НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

«На правах рукопису»
УДК __004.42__

До захисту допущено:
Завідувач кафедри
_____ Сергій Стіренко
«___»_____20__ р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-професійною програмою «Інженерія програмного забезпечення комп'ютерних систем»

зі спеціальності 121 «Інженерія програмного забезпечення»

на тему: «Мобільний додаток для екомоніторингу довкілля»

Виконав:
студент VI курсу, групи ІП-94мп
Гнесь Владислав Олександрович _____

Керівник:
доцент, кандидат технічних наук,
Волокита Артем Миколайович _____

Рецензент:
доцент кафедри АУТС, кандидат технічних наук,
Катін Павло Юрійович _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет (інститут) *Інформатики та обчислювальної техніки*
(повна назва)

Кафедра *Обчислювальної техніки*
(повна назва)

Освітньо-кваліфікаційний ступінь *магістр*
(назва ОКР)

Спеціальність *121. Інженерія програмного забезпечення*
(код і назва)

Спеціалізація *121. Інженерія програмного забезпечення комп'ютерних систем*

ЗАТВЕРДЖУЮ
Завідувач кафедри
Стіренко С.Г.
(підпис) (ініціали, прізвище)
« » 2020 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Гнеся Владислава Олександровича
(прізвище, ім'я, по батькові)

1. Тема дисертації Мобільний додаток для екомоніторингу довкілля

Науковий керівник дисертації доц., к.т.н., Волокита Артем Миколайович
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від ***« 24 » 03 2020 р. № 910-с***

2. Строк подання студентом дисертації

3. Об'єкт дослідження процес екомоніторингу довкілля за допомогою мобільного додатку

4. Предмет дослідження методи та засоби моніторингу екологічної ситуації довкілля на основні технологій для розробки мобільних додатків

5. Перелік завдань, які потрібно розробити:

- Аналіз існуючих рішень у сфері екологічного моніторингу.
- Розробити додаток і забезпечити можливість створення і перегляду екологічних інцидентів в ньому.
- Забезпечити користувачам можливість взаємодії з інтерактивною мапою під час екомоніторингу.
- Розробити стартап-проект для реалізованого програмного продукту.

6. Консультанти розділів дисертації:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормо контроль	Жабін В.І.		

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів дисертації	Примітка
/.	*Затвердження теми роботи*	*10.04.2020-14.05.2020*	
2.	*Вивчення та аналіз завдання*	*15.05.2020-14.06.2020*	
3.	*Розробка архітектури та загальної структури додатку*	*15.06. 2020-10.09.2020*	
4.	*Програмна реалізація додатку*	*10.09. 2020-09.10. 2020*	
5.	*Оформлення пояснювальної записки*	*10.10. 2020-23.11. 2020*	
6.	*Передзахист*	*24.11.2020*	
7.	*Захист*	*16.12.2020*	

Студент _____ _____
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ _____
(підпис) (ініціали, прізвище)

РЕФЕРАТ

на магістерську дисертацію

виконану на тему: Мобільний додаток для екомоніторингу довкілля

студентом: Гнесьом Владиславом Олександровичом

Дипломна робота другого (магістерського) рівня вищої освіти на тему «Мобільний додаток для екомоніторингу довкілля» складається зі вступу та 5 розділів. Загальний обсяг роботи: 118 сторінок, 30 таблиць, 31 рисунок, 1 додаток. Перелік посилань нараховує 30 найменувань.

Актуальність. Актуальність теми дослідження зумовлено необхідністю створення зручних цифрових інструментів для моніторингу екологічної ситуації.

Зв'язок роботи з науковими програмами, планами, темами. Дипломну роботу (магістерського) рівня вищої освіти було виконано в Національному технічному університеті України «Київський політехнічний інститут імені Ігоря Сікорського» відповідно до планів науково-дослідних робіт кафедри обчислювальної техніки.

Метою і завдання дослідження. Метою магістерської роботи є створення зручного кросплатформеного мобільного додатку, що стане платформою для вирішення екологічних проблем.

Для досягнення мети дослідження поставлено і вирішено такі завдання:

- Аналіз існуючих рішень у сфері екологічного моніторингу
- Розробити додаток і забезпечити можливість створення і перегляду екологічних інцидентів в ньому.
- Забезпечити користувачам можливість взаємодії з інтерактивною мапою під час екомоніторингу.
- Розробити стартап-проект для реалізованого програмного продукту.

Об’єкт дослідження – процес екомоніторингу довкілля за допомогою мобільного додатку.

Предмет дослідження - методи та засоби моніторингу екологічної ситуації довкілля на основні технологій для розробки мобільних додатків.

Новизна полягає у тому, що запропоновано мобільний додаток, який відрізняється від існуючих можливістю створення і віслідковування екологічних інцидентів, що дозволяє проводити моніторинг екологічної ситуації навколишнього середовища.

Практична цінність. Розроблений продукт може стати платформою для вирішення соціональний проблем, зокрема екологічного спрямування. Запропоновані підходи до розробки можуть бути використані іншими спеціалістами у існуючих рішеннях або застосувати для розробки нових програмних продуктів.

Ключові слова

Екологічний додаток, платформа, соціальна відповідальність, моніторинг.

ABSTRACT

for master’s degree dissertation

made on the theme: "Mobile application for environmental ecomonitoring"

by student: Hnes Vladyslav Oleksandrovych

Diploma work for the second (master’s degree) level of the higher education on the theme « Protected and fragmentary research system » consists of introduction and 5 parts. Total workload: 118 pages, 30 tables, 31 images, 1 additions. The list of references contains of 30 items.

Topic Relevance. The relevance of research is due to the need to create convenient digital tools for monitoring the environmental situation.

Thesis connection to scientific programs, plans, and topics. The thesis was prepared according to the scientific research plan of Computing Engineering Department of the National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute.”.

Research goal and objectives. The purpose of the master's work is to create a convenient cross-platform mobile application that will become a platform for solving environmental problems.

For reaching the goal the next tasks were set and completed:

- Analyse existing solutions in the field of environmental monitoring.
- Develop an application and provide the ability to create and view environmental incidents in it.
- Provide users with the opportunity to interact with the interactive map during ecomonitoring.
- Develop a start-up project for the implemented software product.

Object of research – the process of ecomonitoring of the environment using a mobile application.

Subject of research – methods and means of monitoring the environmental situation using technologies for the development of mobile applications.

Contribution is that the proposed mobile application, which differs from the existing ones with the ability to create and track environmental incidents, which allows you to monitor the environmental situation.

Practical value. The developed product can become a platform for solving social problems, environmental problems. The proposed approaches to development can be used by other specialists in existing solutions or while developing new software products.

Keywords

Environmental application, platform, social responsibility, monitoring.

ЗМІСТ

ВСТУП

Кожній людині нашої планети важливо усвідомлювати нерозривний зв'язок природи і суспільства, що носить взаємний характер. Зростання масштабів господарської діяльності людини, бурхливий розвиток науково-технічної революції підсилюють негативний вплив на природу і вже призвели до порушення екологічної рівноваги на планеті.

За останні роки потреби в моніторингу екологічних проблем та вирішення хоча б найбільш актуальних з них збільшилась у багато разів.

Екомоніторинг – це спосіб збору та аналізу даних про природні лиха та порушення екологічних норм. З розвитком доступного інтернету та GPS на смартфонах, планшетах та інших портативних пристроях стає легше здійснювати екомоніторинг довкілля власного регіону або країни.

Зрозуміло, що сервіс вирішення екологічних проблем відповідає потребі збереження природних багатств України. Наприклад, мешканці багатьох міст нашої країни постійно зустрічаються із порушенням екологічних норм у роботі великих підприємств. Тому було б зручно, якби вони могли фіксувати факти порушення цих норм та відмічати їх на мапі.

У наш час, використовуючи геолокаційні системи, користувач може отримати безліч інформації про те, де він знаходиться, який шлях подолав,

визначити пройдений маршрут, швидкість з якою рухався, отримати висотні характеристики місцевості тощо. Тому додатки, орієнтовані на отримання подібної інформації, користуються великою популярністю. Через наявність GPS-приймача в більшості сучасних мобільних пристроїв геолокаційна інформація може використовуватися громадянами для зазначення важливих, на їх думку, екологічних проблем. Сервіс моніторингу екологічних проблем може використовуватися для контролю екологічної ситуації на території України і не тільки, а отже тема роботи є актуальною.

РОЗДІЛ 1

ОГЛЯД ТА АНАЛІЗ ОБЛАСТІ ДОСЛІДЖЕННЯ

1.1. Загальна характеристика екологічних проблем в Україні

На початку XXI століття екологічні проблеми набули статусу глобальних. Людство усвідомлює небезпеку скорочення життя на Землі через свій вплив на масштаби природокористування, інтенсивність господарювання, забруднення природного середовища.

Відомо, що екологічні проблеми в тій або іншій мірі завжди супроводжували становлення і розвиток цивілізації. Однак, те, що було в минулому, не може йти ні в яке порівняння з протиріччями, що виникають при взаємодії суспільства і природи в сучасну епоху. Необмежене використання природних ресурсів і вільне викидання відходів у навколишнє середовище призвело до того, що в багатьох країнах практично не залишилося непорушених природних екосистем, спроможних повною мірою виконувати свої функції збереження стану навколишнього середовища. Стійкий розвиток суспільства все більш стримується глобальними екологічними проблемами [1].

Масштаби змін природного середовища залежать від двох основних факторів: інтенсивності прояву речового складу забруднювачів та здатності природи до самоочищення. Остання властивість лежить в основі поняття стійкості окремих компонентів природи до антропогенного навантаження на основі відмінностей у поширенні забруднювачів у різних середовищах. Тверді, рідкі й газоподібні викиди забруднюючих речовин поступають у всі компоненти природи: води, ґрунти, повітря. Найбільше викидів здійснюється в атмосферне повітря, через яке небезпечні речовини поширюються в інші компоненти природи, підвищуючи тим самим уже існуючий в них рівень забруднення [2]. У процесі довготривалої дії забруднювачів погіршуються чи порушуються основні

природні, соціально-економічні функції природного середовища. Це ускладнює життя всіх живих організмів, а особливо людини

Серед найгостріших екологічних проблем України можна виділити:

- неякісну питну воду;
- забруднення повітря;
- деградацію земельних ресурсів;
- знищення лісів;
- небезпечні геологічні проблеми;
- побутові відходи;
- об’єкти військової діяльності;
- наслідки Чорнобильської катастрофи.

Усі ці проблеми потребують оперативного вирішення для виходу України з екологічної кризи. Звісно, мають бути запропоновані різні варіанти для вирішення цієї ситуації [26]. Одним з них може стати застосунок для моніторингу та вирішення екологічних проблем, який завдяки ентузіазму користувачів має привернути увагу влади до висунутого питання.

1.2. Перспективи екологічного стану і пріоритети екологічної політики держави

Аналіз динаміки екологічного стану України показав, що за останні п'ять років екологічна криза продовжує розвиватися, розростається, охоплюючи все більші території України. Цьому сприяють:

- майже повна відсутність асигнувань на серйозні природоохоронні заходи у всіх галузях промисловості;
- відсутність контролю й практична безкарність діяльності, що губить природу;

- зростання кількості та потужності техногенних аварій через тотальне спрацювання устаткування і технологій на виробництвах, а також дуже низька ефективність очисних споруд, з тих, які все ж працюють;
- надзвичайно низький рівень екологічної освіти населення.

Можна, звичайно, дуже довго говорити, про те що необхідно створити черговий контролюючий і каральний орган, який буде вирішувати ці проблеми, але, як показав час, такі заходи не вихід. Люди не перестануть губити природу, тільки тому, що якийсь контролер може їм у цьому перешкодити. Необхідно в першу чергу підвищувати рівень екологічної освіти населення України, і ввести великі матеріальні штрафні санкції за дії, які завдають шкоди навколишньому середовищу, як для промислових підприємств, так і для фізичних осіб [27]. Матеріальна стимуляція таких заходів теж необхідна, але не у вигляді загадкових напрямків бюджетних коштів невідомо на що і невідомо куди, а великі податкові пільги тим підприємствам, які поставлять у себе очисні споруди високої ефективності, і які будуть строго стежити за кількістю стоків, що скидаються. Непрацююча молодь, студенти можуть прекрасно стежити а станом парків, чистотою населених пунктів, якщо їм платити за це. Активна пропаганда в засобах масової інформації про необхідність збереження природних ресурсів і висвітлення реального стану навколишнього середовища в Україні.

Паралельно з цими заходами слід збільшувати площу біосферних заповідників, національних парків, навіть якщо доведеться значно скоротити площу експлуатованих земель. Ми вирощуємо досить, щоб прогодувати себе, продати за кордон, і при цьому зазнаємо величезних збитків, тому що наші землі виснажуються, а допомоги або хоча б відпочинку не отримують.

Покинуті протягом декількох років земельні угіддя, на обробку яких не вистачало грошей змогли хоча б частково самоочиститись і відновитися. Закриті

через економічну нерентабельність великі і середні промислові підприємства перестали забруднювати навколишнє середовище. Жалюгідні залишки Чорноморського флоту дозволили акваторії Чорного моря самоочиститись і підвищити чисельність популяцій деяких вже нечисленних видів морських організмів.

Але це все тільки характеристика, коротко розглянемо ті рекомендації, які можна знайти в західно-європейських книгах з екології.

Альтернативна обробка землі означає відмову від однобокого енергоємного, витратного сільського господарства з інтенсивним застосуванням хімії та повернення цілісного екологічного розгляду сільськогосподарської діяльності. Максимально закритий багатосторонньо структурований робочий цикл повинен будуватися за принципом круговороту речовин в екосистемі між грунтом, рослиною, твариною і людиною. Для цього придатні змішані господарства – вирощування разом рослин і тварин. Застосовується в основному власне і місцева сировина, при цьому необхідно уникати, або мінімізувати навантаження на навколишнє середовище.

Основними положеннями екологічної обробки землі є вирощування фізіологічно повноцінних продуктів харчування і відмова від гербіцидів та інших синтетичних боліт. При такій обробці застосовуються біологічні продукти і методи боротьби з шкідниками в збудниками захворювань. Обов'язково необхідно насаджувати лісопосадки і невеликі гаї. Оскільки проблемою є баланс поживних речовин через збір урожаю, необхідно дуже обережно і точно вивірених кількостях вносити мінеральні добрива [21].

Також в Європі використовуються досить ефективні біологічні методи реставрації озер, очищення стічних вод, фільтрації викидів в атмосферу.

1.3. Моніторинг навколишнього природного середовища

У природному середовищі проводиться екологічний моніторинг, який дозволяє спостерігати динаміку змін всіх процесів в екосистемах. Всі дані збираються спеціальними службами з різних об'єктів, проводяться спостереження, за якими в подальшому полягає аналіз.

Екомоніторинг за ступенем досліджень і масштабності розділяється на:

- біоекологічний, на якому аналізуються санітарно-гігієнічні норми;
- геосистемний, в ході якого вивчаються дані господарських і природних угідь;
- біосферний, для якого складається загальна картина в масштабах планети.

Для моніторингу стану навколишнього середовища збираються різні дані по рівню забруднення повітря і водойм, показники погоди і стан неживої природи. Також здійснюється дослідження всіх кліматичних даних і змін. На рівні біологічного моніторингу проводиться спостереження за живими організмами і їхнім станом при забрудненні і зміни навколишнього середовища. Крім того, в екомоніторинг входить збір даних захворюваності та стану здоров'я людей. Все це дозволяє прогнозувати стан земної біосфери і виявляти екопроблеми.

В цілому збір даних проводиться на різних рівнях:

- детальний – дослідження невеликого земельної ділянки або території;
- локальний – проводиться в рамках району або населеного пункту;
- регіональний – вивчається стан обласного рівня;
- національний – здійснюється екомоніторинг конкретної країни;
- глобальний – проводиться в рамках програми ООН, вивчається зміни в планетарному масштабі.

Екологічний моніторинг проводиться на постійній основі спеціальними відомствами [22]. Ця інформація дозволяє отримати дані на рахунок стану навколишнього середовища в певний час з максимальною точністю, щоб проводити очищення біосфери і раціонально витрачати природні ресурси [3]. Також це дозволяє спостерігати за кругообігом речовин в середовищі, визначати час розкладання відходів різного виду, утилізувати деякі з них і знизити антропогенний вплив на природу, щоб долати різні екологічні проблеми.

Таким чином, моніторинг навколишнього середовища є невід'ємною діяльністю, щоб контролювати стан нашої планети. Він дозволяє своєчасно фіксувати всі зміни, на основі яких будується прогноз. У свою чергу, він допомагає визначити, як потрібно витрачати ті чи інші природні блага.

Під програмою системи моніторингу розуміють сукупність цілей організації, конкретних стратегій поведінки і механізмів реалізації. Основними складовими є:

- об'єкти, що мають територіальну прив'язку, які знаходяться під строгим контролем служб;
- показники контролю;
- допустимі області зміни показників;
- часові масштаби.

Кожна програма містить розроблені карти, таблиці, в яких вказані місця дати, а також методи відбору проб, схеми та інші важливі дані. Також в програму закладені методи дистанційного аналізу, що дозволяють визначити стан навколишнього середовища.

1.4. Порівняння наявних систем моніторингу громадських проблем

Перед початком виконання дипломної роботи було проведене детальне дослідження предметної області та існуючих рішень. На даних момент, є дкілька прикладів програмного забезпечення та застосунків, які певним чином пов’язані з екомоніторингом та вирішення соціально важливих проблем. Тож, давайте розглянемо існуючі аналоги.

1.4.1. Програмний продукт «СЕМОС»

Система екологічного моніторингу навколишнього середовища виконує неперервний автоматизований моніторинг за станом навколишнього середовища та забезпечує своєчасне інформування відповідальних осіб достовірною інформацією для прийняття ефективних управлінських рішень в області природоохоронної діяльності та моніторингу забруднення [30]. Дана система дозволяє:

- неперервно в цілодобовому режимі здійснювати моніторинг рівня забруднення атмосферного повітря, водного басейну, ґрунтового покриву на підконтрольній території;
- встановити географічне розташування джерел забруднення та оцінити їх вплив на поточну екологічну обстановку (формування профіля викидів);
- використовувати режими повідомлення та попередження про аварійні ситуації, а також про можливе підвищення забрудненості у зв’язку з несприятливими метеорологічними умовами;

Інтерфейс користувача програми «СЕМОС» показаний на рис 1.1.

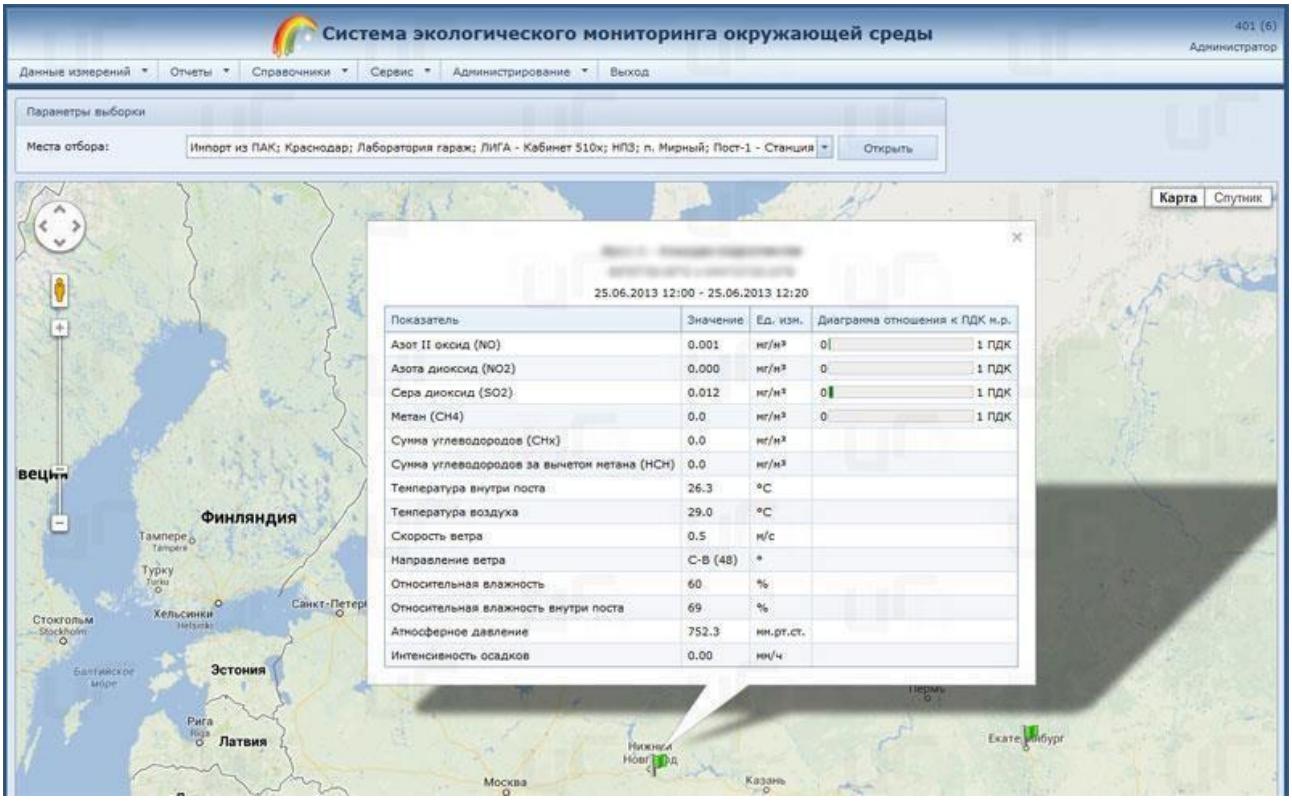

Рис. 1.1. Інтерфейс програми «СЕМОС»

Отримання інформації здійснюється шляхом передачі даних від газоаналізаторів та аналітичних пристроїв (що знаходяться в стаціонарних та рухомих лабораторіях) та зведенні до мінімуму “людського” фактору на всіх етапах отримання та обробки інформації [4]. Але додаток є російськомовним.

1.4.2. Застосунок «Каратєль»

Ще одним прикладом геомоніторингу є український застосунок «Каратєль». З його допомогою можна анонімно поскаржитися майже на будь-яке правопорушення: вимагання хабаря, хамство, порушення правил паркування, продаж товару з порушеним строком дії, паління у закладах харчування, порушення благоустрою, ями на дорогах тощо [5]. Інтерфейс користувача додатку «Каратєль» показаний на рис 1.2.

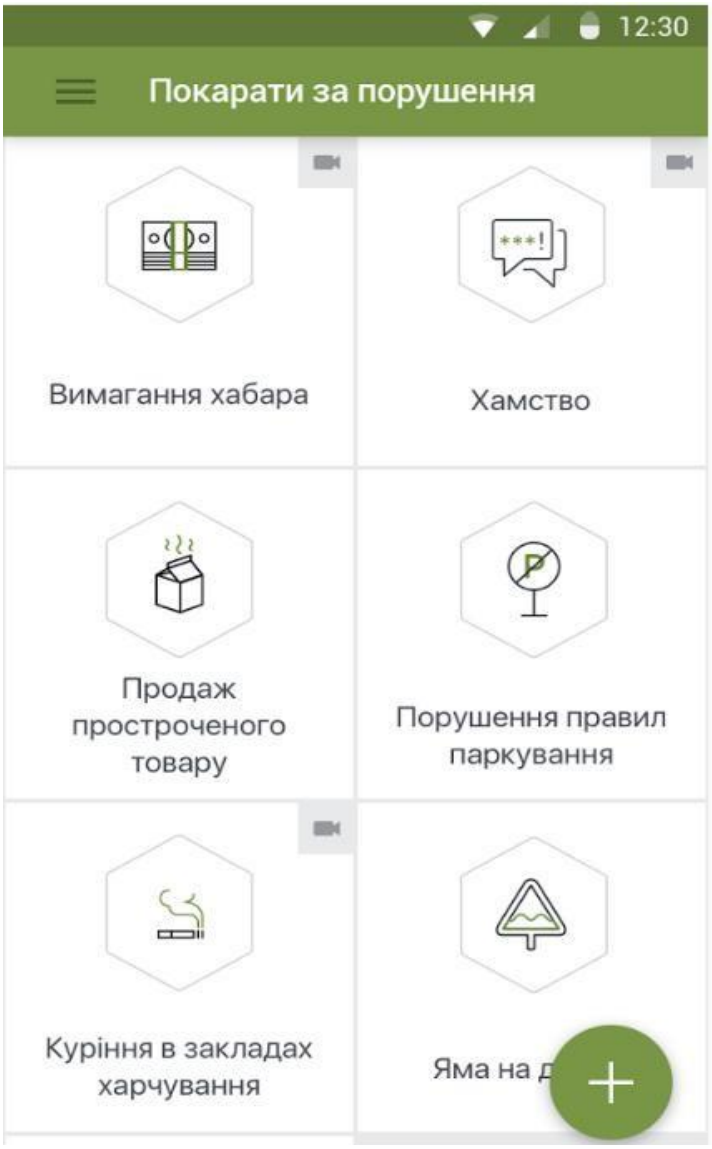

Рис. 1.2. Інтерфейс додатку «Каратєль»

Юристи проекту оброблять заявку та відправлять звернення до відповідного органу державної влади, які будуть зобов’язані це правопорушення усунути [25]. Відповідь користувач зможе подивитися в додатку на своєму мобільному телефоні. При цьому від користувача, який відправив скаргу, не вимагається платня за послуги юристів.

Не дивлячись на те, що скарга користувача є анонімною, організатори проекту все ж просять зареєструватися та вказати максимально точно свої персональні дані. Це необхідно для того, щоб команда розуміла, що скарга подана реальною людиною та надана інформація відповідає дійсності. Можливо, юристам знадобиться періодично зв’язуватися з користувачем, щоб підтвердити подану інформацію та з’ясувати деталі[5]. Але при цьому особисті дані користувача будуть доступні лише обмеженому колу співробітників ГО “Фундація.101” та не фігурують у зверненнях до органів влади або приватних структур.

1.4.3. Додаток «Eyewitness»

Цей додаток розроблений британською компанією. Останнім часом розглядається можливість використання його у діяльності Національної поліції України [29]. За допомогою цієї програми кожен громадянин матиме можливість сфотографувати правопорушення на свій гаджет та миттєво відправити фото з коротким описом події до правоохоронних органів.

Інтерфейс користувача додатку «Eyewitness» показаний на рис 1.3.

Рис. 1.3. Інтерфейс додатку «Eyewitness»

Громадянин, який сповістив про скоєне правопорушення, може, за бажанням, залишатися анонімним навіть під час контакту з представниками правоохоронних органів [6]. Відмітимо, що хоч інформацію, отриману за допомогою цієї програми, і не можна використовувати як доказ у судовому засіданні, її можна використати для оперативного реагування підрозділів.

1.4.4. Додаток «TrashOut»

Екологічний мобільний додаток TrashOut – це перша інтерактивна карта несанкціонованих звалищ для iOS і Android, за допомогою якої можна відзначати на карті місця зі сміттям, інтерфейс зображено на рис. 1.4. Додаток, створений словацькими розробниками тепер актуальний в Україні.

Безкоштовний додаток для iOS і Android тепер є і українською мовою, завдяки партнерству TrashOut і екологічної ініціативи «Смарагдова планета». Таким чином, Україна приєднується до глобальної екологічної ініціативи.

«TrashOut – більш ніж актуальний додаток для України, адже в нашій країні ситуація з твердими побутовими відходами (ТПВ) катастрофічна: більше 90% сміття вивозиться на звалища, за оцінками Укрприроднагляду, їх на сьогоднішній день більше 35 000, абсолютна більшість з них є несанкціонованими [7]. Все це завдає непоправної шкоди навколишньому середовищу і здоров'ю українців. За допомогою програми TrashOut і активних громадян ми дізнаємося про реальні місця розташування як великих, так і невеликих несанкціонованих звалищ. А найголовніше – наочно продемонструємо владі і всім українцям масштаби проблеми і будемо вимагати екстрених заходів для усунення конкретних звалищ.

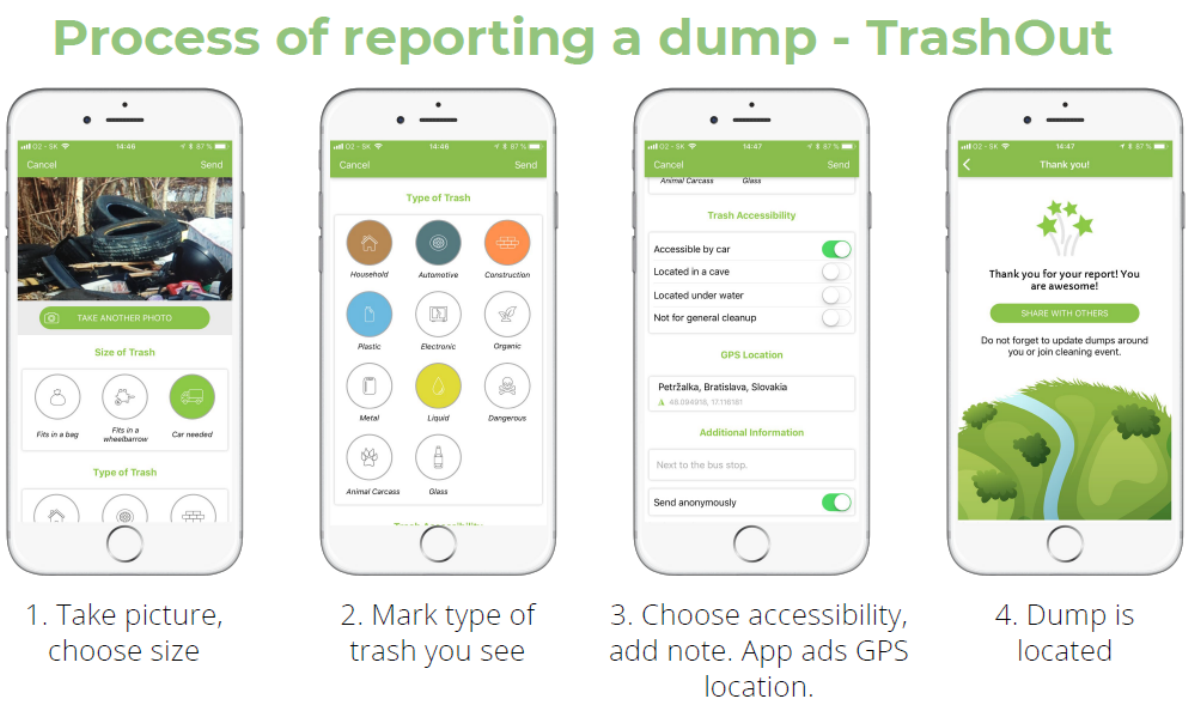

Рис. 1.3. Інтерфейс додатку «TrashOut»

Щоб взяти участь в складанні світової карти несанкціонованих звалищ необхідно:

1. Встановити додаток на мобільний телефон. Додаток TrashOut доступно для iPhone і телефонів з операційною системою Android.
2. Відзначати звалища на карті.
3. Якщо ви знайшли несанкціоноване звалище, сфотографуйте його за допомогою програми TrashOut, виберіть її розмір і тип переважаючих відходів, і нова позначка з'явиться на мапі сміття. Додаток автоматично визначає місце розташування звалища.

Дані інтерактивної карти можуть бути використані для складання маршрутів ліквідації сміттєзвалищ, в рамках програм які проводять державні органи і екологічні організації.

За допомогою цієї програми можна визначити місцезнаходження незаконного звалища в вашому районі, місті або в будь-якій точці світу, і

повідомити про нього спільноті. Проект дозволяє кожному користувачеві зробити свій внесок в роботу з поліпшення стану навколишнього середовища. Дані, зібрані користувачами TrashOut, допоможуть місцевим інститутам, організаціям та урядам виправити екологічну ситуацію в світі.

Розробники програми TrashOut – молоді еко-активісти зі Словаччини, які вирішили створити ефективний інструмент, який дозволить визначати, фіксувати і актуалізувати точне місце розташування несанкціонованих звалищ на світовій карті за допомогою мобільного телефону. На початку 2013 року додаток TrashOut увійшло в число фіналістів Європейського Кубка кращих додатків для мобільних телефонів (European AppCup).

1.5. Постановка задачі

Було поставлено за мету створити мобільний застосунок, який буде здійснювати моніторинг та вирішення основних екологічних проблем України. Основною функцією клієнтської частини буде можливість оформити скаргу на порушення екологічної безпеки навколишнього середовища (користувач зможе обирати з основних категорій найбільш розповсюджених екологічних правопорушень). Небайдужий громадянин матиме змогу вказати місце правопорушення на мапі, визначити найближчий до цього місця орієнтир та детально описати те, що трапилося.

Таблиця 1.1 – Порівняльна таблиця додатків

Функціонал	CEMOC	Каратєль	Eyewitness	TrashOut	Розроблений додаток
Створення інциденту	-	+	+	+	+

Можливість моніторингу екологічних проблем	+	-	-	+	+
Можливість оперування мапою в додатку	+	-	-	-	+
Можливість створення власних екологічних інцидентів, використовуючи фото і геолокаційні дані	-	-	-	-	+
Кросплатформеність	-	+	-	+	+

Було вирішено використовувати збереження інформації до бази даних для автоматизації процесу геомоніторингу.

Обидва пристрої мають працювати під керуванням мобільної ОС Android або iOS та підтримувати датчик GPS. Застосунок має стабільно працювати на усіх останніх версіях Android і iOS. Порівняльну характеристуку додатків можна побачити на таблиці 1.1.

Висновки до розділу 1

Впродовдж ознаймолення та огляду програмних продуктів, які реалізовують схожий функціонал було оглянуто основні продукти, які є аналогами до розробленого рішення:

- СЕМОС
- Каратєль
- Eyewitness.
- TrashOut

Варто відмітити, що на відміну від аналогів, даний додаток буде здійснювати саме екологічний моніторинг і привертатиме увагу суспільства до проблем нашої держави. Виходячи з цього можна заявляти що розроблений програмний продукт не буде мати прямих аналогів.

У ході знайомства та огляду вищевказаних програмних продуктів, які реалізовують схожий функціонал, було виявлено важливі недоліки цих систем:

- Немає одного спеціалізованого додатку для вирішення екологічних проблем;
- Відсутність оперування інтерактивною мапою в більшості аналогів;
- Відсутність можливості створення власних екологічних інцидентів, використовуючи фото і геолокаційні дані.

Після тривалого та детального аналізу переваг та недоліків аналогів існуючих продуктів було прийнято рішення розробити власну систему для екомоніторингу навколишнього середовища.

РОЗДІЛ 2

ОПИС ІНСТРУМЕНІВ ДЛЯ РЕАЛІЗАЦІЇ ТА АРХІТЕКТУРИ ДОДАТКУ

2.1. Вибір мови і технологічної платформи для розробки мобільного додатку.

Для реалізації додатку я обрав мови програмування Dart і Java. Для реалізації клієнтської частини був використаний фреймворк Flutter. Для реалізації серверної частини було використаний Spring Framework та його підмодулі Spring Boot, Security, Cloud, Data. Для збереження даних було використано базу даних PostgreSQL. Для зручного розвертання мікросервісів і бази даних було використано середовище Docker.

2.1.1 Мова програмування Dart.

Dart - це клієнто-оптимізована мова програмування для розробки додатків на різних платформах. Вона розроблена компанією Google і використовується для створення мобільних, десктопних, серверних та веб-додатків.

Рис. 2.1. Логотип мови програмування Dart

Логотип мови програмування Dart можна побачити на рис. 2.1.

Dart - це об'єктно-орієнтована мова програмування яка базується на синатиксисі мови C, що використовує збирач сміття, як модель для управління пам`яттю. Розробники мови позіціонують її як мову, що у майбутньому поступово замінить мову JavaScript, яка має досить багато проблем що зв`язані з продуктивністю, підтримкою і розробкою складних клієнтських застосунків.

2.1.2 Фреймворк Flutter.

Flutter - це фреймворк з відкритим кодом для створення високоякісних, високопродуктивних мобільних додатків у мобільних операційних системах - Android та iOS. Він надає простий, потужний, ефективний і зрозумілий SDK для написання мобільних додатків власною мовою Google, Dart.

Загалом, розробка мобільного додатка є складним та об`ємним завданням. Існує багато фреймворків для розробки мобільних додатків. Android дає змогу розробляти використовуючи нативний фреймворк на основі мови Java, а iOS – рідний фреймворк на основі мови Objective-C / Swift. Схему роботи Flutter можна побачити на рис. 2.2.

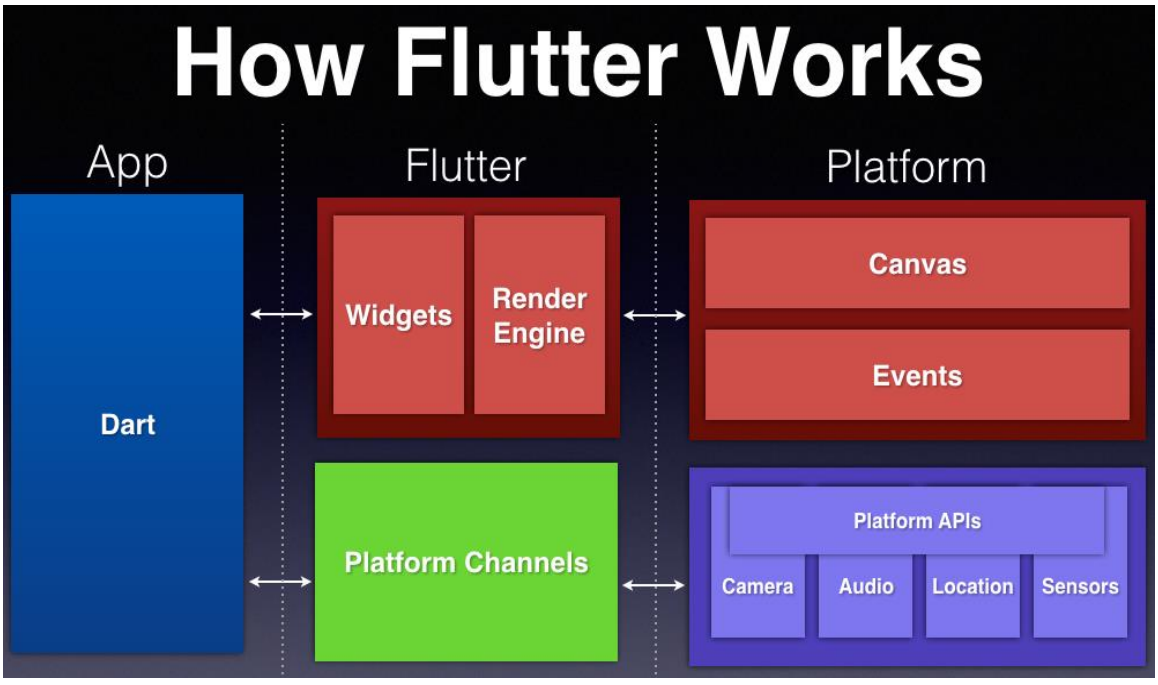

Рис. 2.2. Схема роботи фреймворку Flutter

Однак, щоб розробити додаток, що який будуть підтримувати обидві ОС, нам потрібно написати код двома різними мовами з використанням двох різних фреймворків. Щоб допомогти подолати цю складність, існує багато фреймворків, що дозволяють писати мобільний додаток який підтримується обома ОС. Ці

фреймворки варіюються від простих гібридних фреймворків на базі HTML (які використовують HTML для інтерфейсу користувача та JavaScript для логіки додатків) до складних фреймворків зав`язаних на конкрутній мові програмування (що роблять важку роботу перетворення коду з певної мови програмування у нативний код). Незалежно від їх простоти або складності, ці фреймворки завжди мають багато недоліків, одним з головних недоліків завичай є їх повільна продуктивність.

У цьому випадку Flutter - простий і високопродуктивний фреймворк, заснований на мові Dart, забезпечує високу продуктивність, надаючи інтерфейс користувача безпосередньо через методи операційної системи, а не через нативний код.

Flutter також пропонує багато готових віджетів (UI) для створення сучасного додатка. Ці віджети оптимізовані для мобільного середовища, а розробка програми за допомогою віджетів проста, як і розробка за допомогою HTML.

Якщо бути конкретним, програма Flutter сама по собі є віджетом. Віджети Flutter також підтримують анімацію та жести [10]. Логіка програми заснована на реактивному програмуванні. Віджет за бажанням може мати стан. Змінюючи стан віджета, Flutter автоматично (реактивне програмування) порівнює стан віджета (старий та новий) та відображатиме віджет лише з необхідними змінами замість того, щоб повторно відтворювати весь віджет.

Переваги Flutter:

1. Dart має велике сховище програмних пакетів, що дозволяє розширити можливості вашої програми.
2. Розробникам потрібно написати лише код для обох програм (як для платформ Android, так і для iOS). Flutter може бути доступним і на інших платформах в майбутньому.

3. Flutter потребує менше тестування. Через єдину базу коду достатньо, якщо ми напишемо автоматизовані тести один раз для обох платформ.
4. Flutter потребує менше тестування. Через єдину базу коду достатньо, якщо ми напишемо автоматизовані тести один раз для обох платформ.
5. За допомогою Flutter розробники мають повний контроль над віджетами та їх макетом.
6. Flutter пропонує зручні інструменти для розробників із гарячим перезавантаженням.

2.1.3 Фреймворк Spring Framework.

Spring Framework - це фреймворк додатків та контейнер інверсії контролю для платформи Java. Він є де-факто стандартом у розробці сучасних веб додатків і забезпечує розробника великим вибором опцій для реалізації будь-яких потреб. Екосистему, що складаться з багатьох спеціалізованих сервісів можна побачити на рис. 2.3.

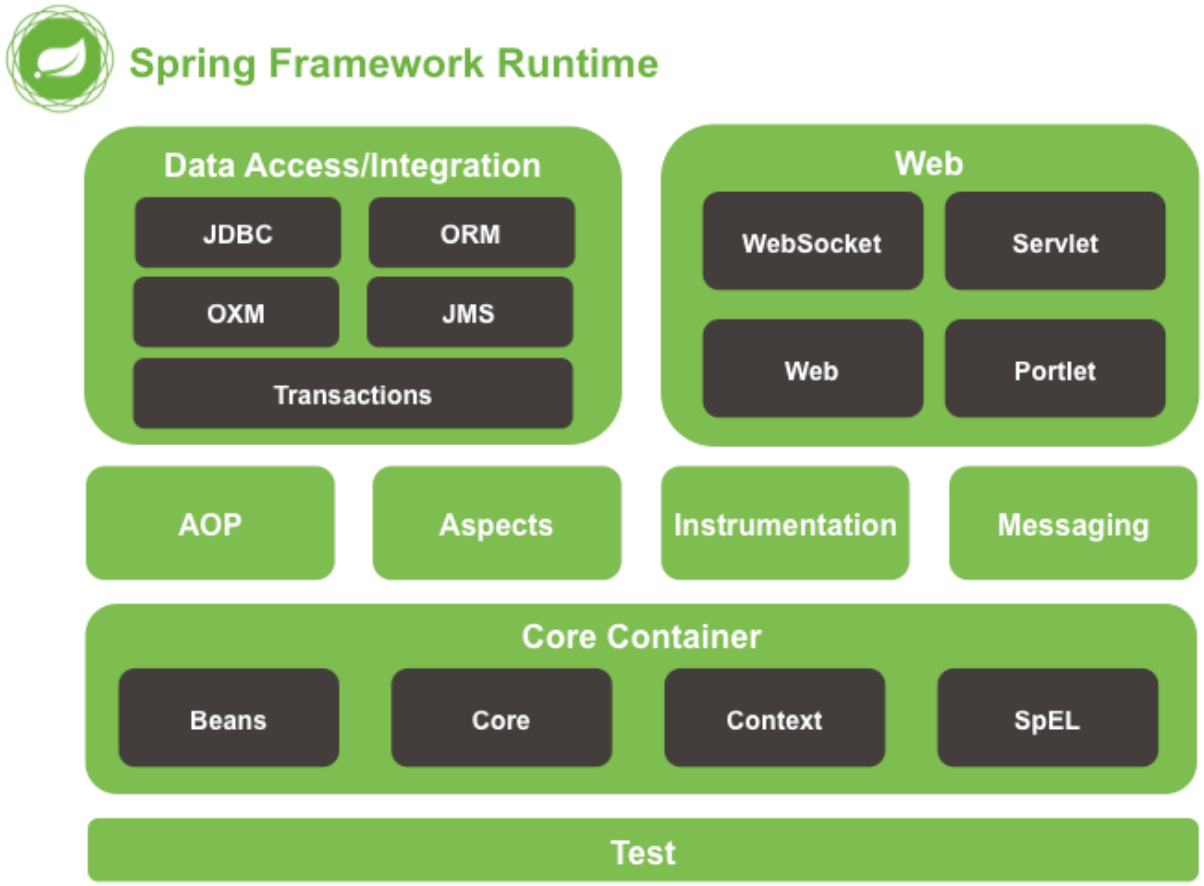

Рис. 2.3. Екосистема Spring Framework

Spring має цілу низку дочірніх підпроектів що дають змогу вирішити майже будь-яку проблему при розробці сучасних веб додатків. Хоч Spring і є модульним, він дозволяє вам вибрати, які модулі вам підходять, без необхідності завантажувати решту.

Spring Boot - це фреймворк на основі Java з відкритим кодом, який використовується для створення мікросервісів. Він розроблений Pivotal Team як дочірній підпроект Spring. Логотип Spring Boot зображено на рис 2.4.

Рис.2.4. Логотип Spring Boot

За допомогою Spring Boot легко створити окремі та готові Spring програми. Spring Boot містить всебічну підтримку інфраструктури для розробки мікросервісів та дозволяє розробляти готові до використання програми, які ви можете «просто запустити». Крім того Spring Boot містить у собі вбудований контейнер сервлетів, що дозволяє з легкістю розгортати магато незалежних мікросервісів.

Ще одним проектом який використовуєть у даній роботі є Spring Data, а якщо конкретніше, то Spring Data JPA. Це підпроект з екосистеми Spring, що реалізує Java Persistence Api і надає швидкий та гнучкий спосіб для роботи з реляційними базами даних [15]. Написання коду для доступу до даних у реляційній базі даних було досить непростим завданням тривалий час. Схему роботи Spring Data JPA можна побачити на рис. 2.5. Для виконання простих запитів, а також виконання пагінації та аудиту потрібно написати занадто багато типового коду. Spring Data JPA значно покращує реалізацію рівнів доступу до даних, зменшивши зусилля в кілька разів.

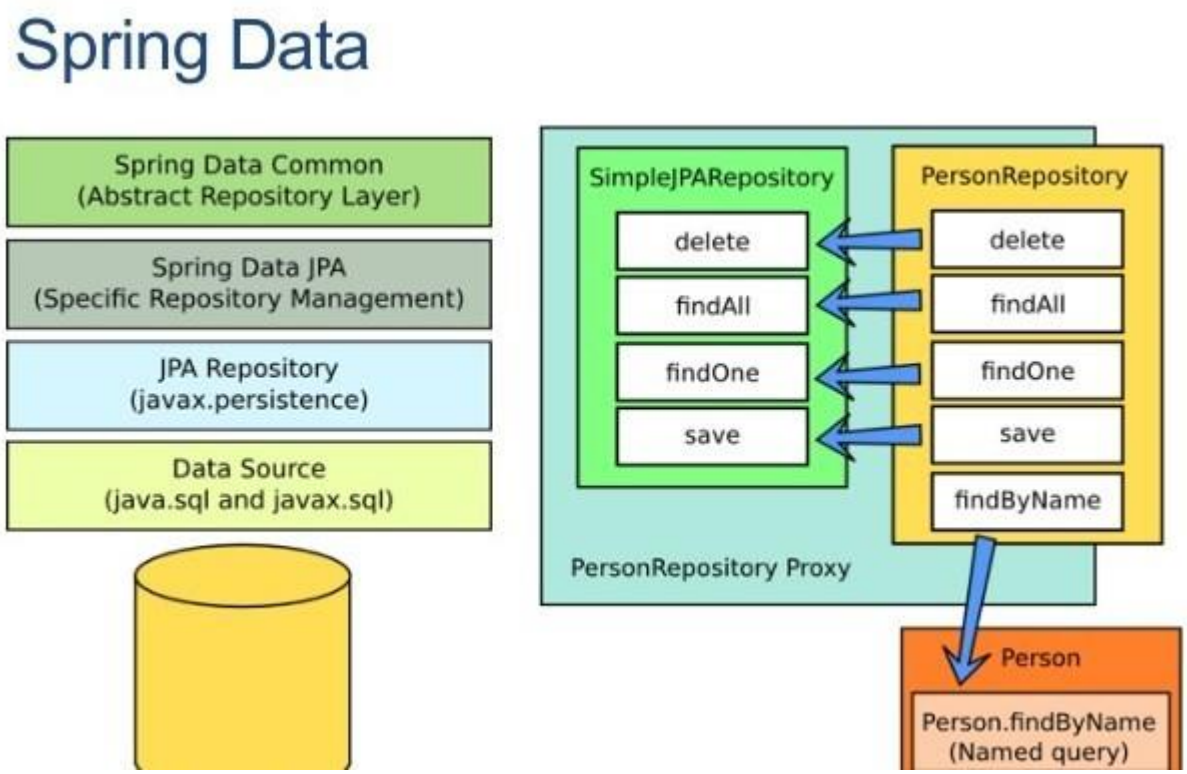

Рис.2.5. Схема роботи Spring Data JPA

Ще однією значною складовою будь-якого додатку є авторизація і можливість простої реалізації системи безпеки. Для цього використовується і ще один продукт з низки підпроектів екосистеми Spring Framework, а саме Spring Security.

Spring Security - це потужна та налаштовувана система автентифікації та контролю доступу. Це фактичний стандарт захисту програм на основі Spring. Spring Security – це фреймворк, який зосереджений на забезпеченні автентифікації та авторизації програм Java. Як і всі проекти Spring, справжня сила Spring Security полягає в тому, як легко її можна розширити, щоб задовольнити власні потреби. Крім того за допомогою Spring Security можна доволі просто реалізувати протокол OAuth або побудувати систему авторизацію побудовану на JWT(Javascript Web Token).

Spring Cloud надає розробникам інструменти для швидкого побудови деяких загальновживаних моделей для розподілених систем. Для наших потреб ми будемо використовувати OpenFeign який є частиною Spring Cloud.

Open Feign – є декларативним веб клієнтом, що легко інтегрується в екосистему Spring. Фактично, Feign створює динамічну реалізацію інтерфейсу, помічену анотаціями JAX-RS або Spring MVC. Чим забезпечує надійний, простий для конфігурації спосіб для комунікації між мікросервісами.

2.1.4 PostgreSQL

PostgreSQL, також відомий як Postgres, - це безкоштовна реляційна система управління базами даних із відкритим кодом (RDBMS), що дозволює розширюваність та є SQL-відповідною. Приклад реалізованої бази даних PostgreSQL для нашого додатку у клієнті DBeaver зображений на рис.2.6.

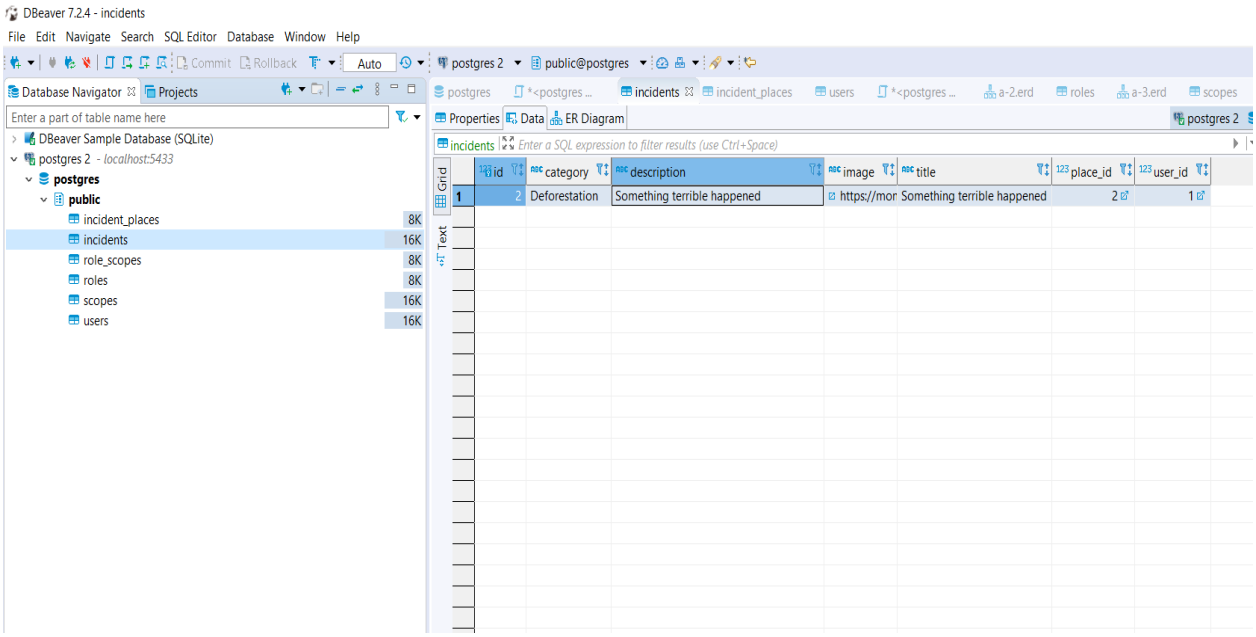

Рис.2.6. Вигляд реалізованої бази даних PostgreSQL для нашого додатку у клієнті DBeaver

Спочатку вона називалася POSTGRES, посилаючись на своє походження як наступник бази даних Ingres, розробленої в Каліфорнійському університеті, Берклі. Є альтернативою як комерційним, так і системам керування базами даних з відкритим кодом [16].

PostgreSQL реалізує транзакції зі властивостями Atomicity, Consistentity, Isolation, Durability (ACID), автоматично оновлювані розрізи даних(View), матеріалізовані розрізи даних, тригери, зовнішні ключі та збережені процедури. PostgreSQL заслужив сильну репутацію завдяки своїй перевіреній архітектурі, надійності, цілісності даних, надійному набору функцій, розширюваності та відданості open-source спільноті, що стоїть за програмним забезпеченням, для постійного забезпечення продуктивних та інноваційних рішень.

Чому була обрана система управління базами даних PostgreSQL?

- **Типи даних.** Усі необхідні типи даних підтримуються в PostgreSQL, такі як примітиви (ціле число, число, рядок, булеве значення тощо), структуровані (дата / час, позначка часу, інтервал, масив, діапазон, uuid, перелік тощо), документи (JSON, XML, Hstore тощо), геометрія (точка, лінія, коло, багатокутник тощо).
- **Цілісність даних.** Найбільш необхідні обмеження підтримуються в PostgreSQL, такі як UNIQUE, NOT NULL, Primary Keys, Foreign Keys та Exclusions. Ви можете використовувати явне блокування (підтримується ACID), консультативні блокування.
- **Нереляційні дані.** Є можливість використовувати власні типи даних документів (JSON, XML, Hstore та Cstore), щоб перетворити PostgreSQL у базу даних NoSQL. Зміст цих типів даних можна проіндексувати, забезпечуючи велику швидкість та цілісність даних.

2.1.5 Docker

Docker - це інструмент щоб був спроектований для спрощення створення, розгортання та запуску програм за допомогою контейнерів [9]. Контейнери дозволяють розробнику упакувати програму з усіма необхідними частинами, такими як бібліотеки та інші залежності, та розгорнути її як один пакет. Завдяки цьому, розробник може бути впевнений, що програма працюватиме на будь-якій іншій машині Linux, незалежно від будь-яких налаштованих параметрів, які можуть відрізнятися від параметрів машини, яка використовується для написання та тестування коду.

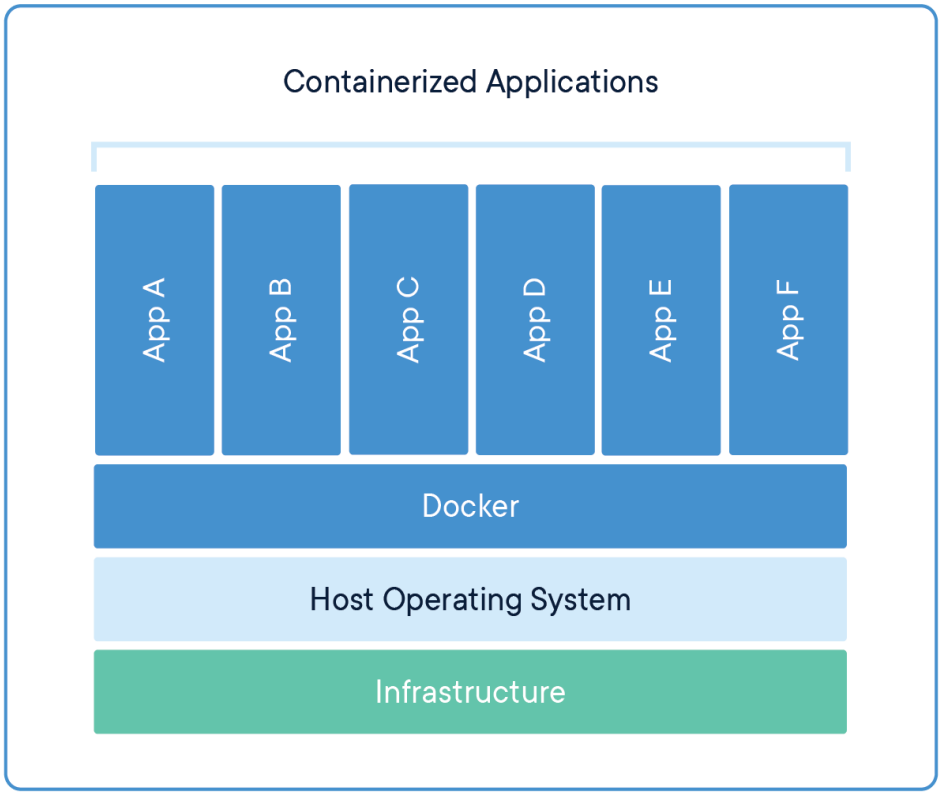

Рис.2.7. Схема роботи контейнеризованого додатку

Певним чином, Docker трохи схожий на віртуальну машину. Але на відміну від віртуальної машини, замість створення цілої віртуальної операційної системи, Docker дозволяє програмам використовувати те саме ядро Linux, що і система, в

якій вони працюють. Це дає значний приріст продуктивності та зменшує розмір програми.

2.1.6 Google Maps

Для роботи з інтерактивною мапою було вирішено використовувати сервіс, що Google Maps, що доступний на хмарній платформі Google Cloud Platform.

Google Maps - це веб-служба картографування, розроблена Google. Вона пропонує платформу для доступу до супутникових знімків, аерофотозйомки, карти вулиць, інтерактивного панорамного виду вулиць в режимі реального часу, також дозволяє планування маршруту для пішохідних поїздок, автомобілів, велосипедів, та громадського транспорту.

Для реалізації потрібного функціоналу було використано наступні API:

- Geocoding API
- Maps Static API
- Places API

Geocoding API дозволяє використовувати пряме і зворотне геокодування. Геокодування - це процес перетворення адрес (наприклад, "Politekhnichna St, 37, Kyiv, Ukraine, 02000") у географічні координати (наприклад, широту 37,423021 та довготу -122,083739), за допомогою яких можна розміщувати маркери на карті. Зворотне геокодування - це процес перетворення географічних координат у зручну для читання адресу. Ви також можете використовувати Geocoding API, щоб знайти адресу для даного певного місця, використовуючи його Place Id. Geocoding API забезпечує прямий спосіб доступу до цих служб через HTTP-запит [11].

Maps Static API дозволяє вставляти зображення Google Maps на свою веб-сторінку, не вимагаючи JavaScript або будь-якого динамічного завантаження сторінки. Послуга Maps Static API створює вашу карту на основі параметрів URL-адреси, надісланих через стандартний запит HTTP, і повертає карту як зображення, яке ви можете відобразити на своїй веб-сторінці або у нашому випадку, на екрані

додатку [12]. Приклад використання Maps Static API у реалізованому додатку можна побачити на рис 2.8.

Рис.2.8. Приклад використання Maps Static API у реалізованому додатку

Places API - це служба, яка повертає інформацію про місця, що використовують HTTP-запити [14]. Місця в цьому API визначаються як заклади, географічне розташування або визначні пам'ятки. Ця служба отримує доступ через HTTP-запит і повертає відповідь у JSON або XML. Усі запити до служби Places повинні використовувати протокол https: // та включати ключ API [13].

Places API використовує ідентифікатор місця для однозначної ідентифікації місця – Place id.

2.2. Загальний огляд архітектури додатку

Для розробки мобільного додатку для екомоніторингу довкілля було обрано клієнт-серверну архітектуру, де клієнтом у нас виступає додаток для мобільних платформ з операційною системою Android або iOS розроблений на мові програмування Dart, в свою чергу, серверна частина реалізована засобами мови програмування Java. Взаємодія між елементами системи здійснюється за допомогою протоколу HTTP.

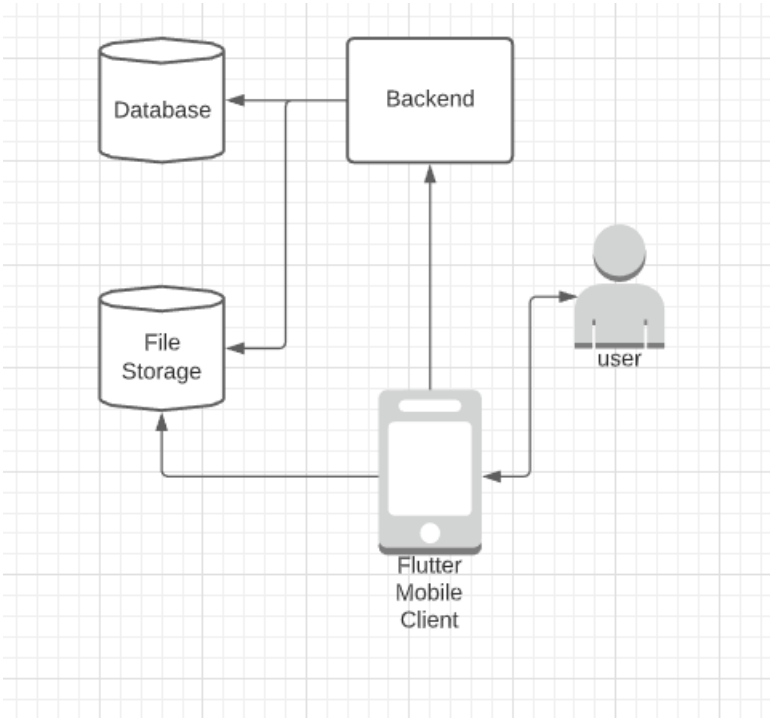

Рис.3.1. Загальна архітектура додатку

Модель нашого додатку можна умовно поділити на 4 частини, а саме:

- мобільний клієнт
- база для зберігання файлів
- серверна частина

- реляційна база даних

Для реалізації серверної частини додатку було вибрано мікросервісну архітектуру. Це такий підхід до розробки програмних продуктів, де розробник розбиває програмні компоненти на ізольовані, самодостатні сервіси, що взаємодію між собою за допомогою певних протоколів передачі даних. Варто відмити, що слід відповідально підходити до питання розбивання продукту на окремі сервіси, орієнтуючись при цьому на їхню бізнес цінність та атомарність.

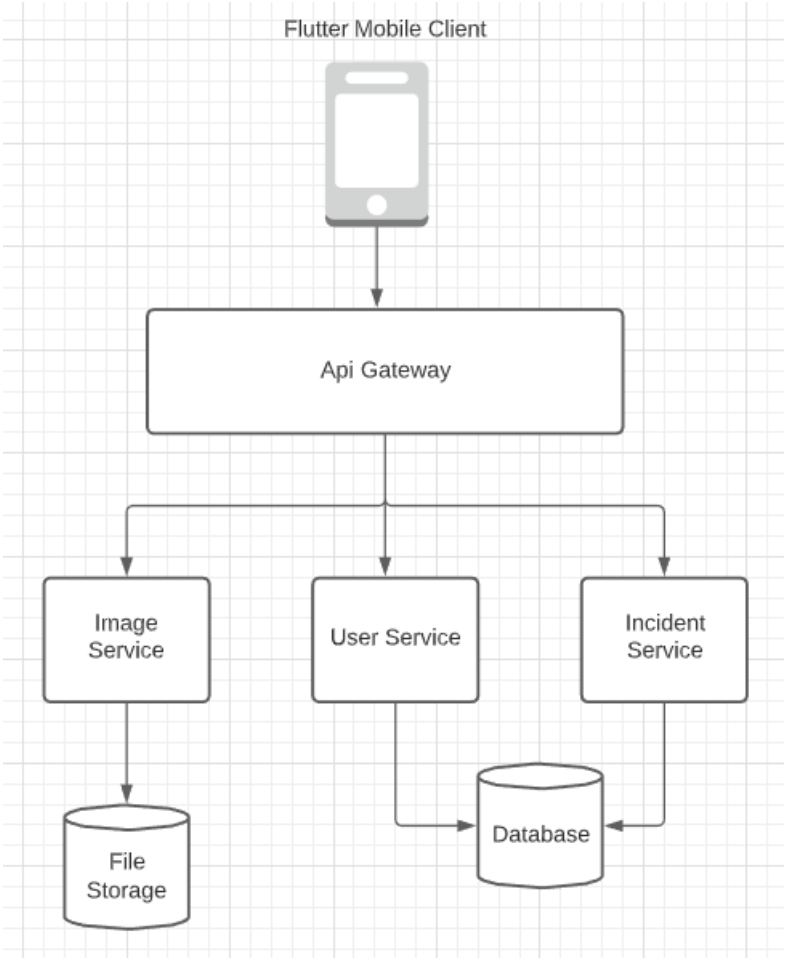

Рис. 3.2. Архітектура серверної частини додатку

У випадку нашого додатку було виділено основі три сервіси для роботи з різними логічними частинами додатку. А саме:

- Image service – сервіс, що відповідає за обробку і видачку картинок та їх збереження у базі для збереження картинок
- User service – сервіс, що відповідає за профіль користувача у системі. Обробку, видачу та збереження даних що стосуються користувача.
- Incident service – сервіс, що відповідає за інцидент екологічних порушень у системі. Обробку, видачу та збереження даних що стосуються інцидентів.

Ще однією складовою архітектури серверної частини є Api Gateway сервіс, що є вхідною точкою для запитів з клієнта і відповідає за координацію запитів до інший сервісів задля формування єдиної відповіді клієнту [17].

Також для кращого розділення бізнес значення від інших процесів, сервіси реалізовують багатошарову архітектуру.

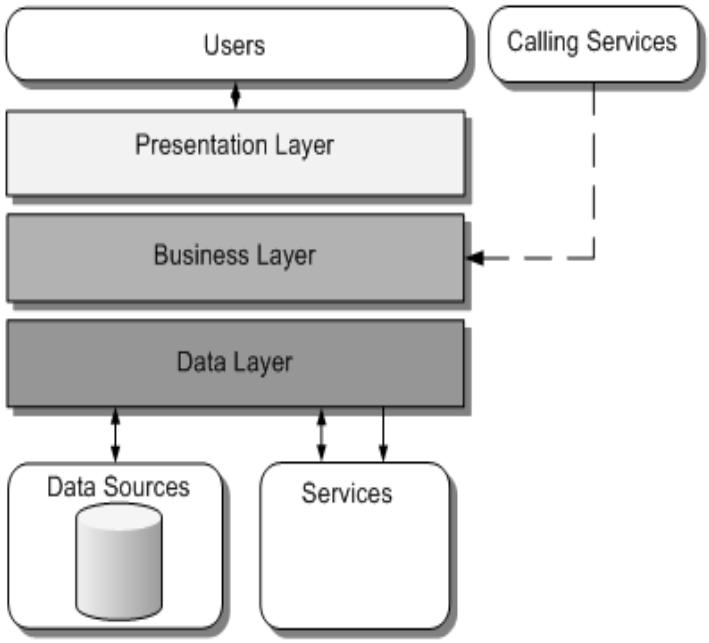

Рис. 3.3. Архітектура серверної частини додатку

- **Рівень представлення(Presentation Layer)** – відповідає за обробку і представлення бізнес сутностей і компонетів.

- **Рівень бізнес логіки (Business Layer)** – скаладається з логіки, бізнес сутностей та їх компонентів
- **Рівень даних(Data layer)** – включає в собі утилитні класи для роботи з даними, забезпучує доступ до бази даних.

Висновки до розділу 2

Отже, у даному розділі було оглянуто основні інструменти та архітектурні приципи, що будуть потрібними про розробці додатку для екомоніторингу довкілля. Умовно вибрані інструменти можна поділити на дві частини: потрібні для розробки серверної частини та потрібні для розробки клієнтської частини.

Як основу для розробки клієнтської частин було обрано мову програмування Dart, що дозволяє розробку кросплатформених рішень для мобільних платформ і фреймвор Flutter, що дозволяє створювати графічний інтерфейс користувача за допомогою віджетів. Для реалізації функціоналу, що пов’язаний з операціями з мапою було обране рішення від Google Cloud Platform – Google Maps.

Для розробки серверної частини було обрано мову програмування Java. В якості фреймворка, що значно полегшить розробку було вибрано Spring, а саме такі проекти що входять до екосистеми Spring - Spring Boot, Spring Data JPA, Spring Security, Spring Cloud. В якості бази даних було вибрано популярну базу даних з відкритим вихідним кодом PostgreSQL.

Також у цьому розділі було розглянуто основні архітектурні принципи що були використані для реалізації клієнтської і серверної частин додатку. Вибрано мікросервісний архітектурний підхід і розділено предменту область на декілька сервісів.

РОЗДІЛ 3

РОЗРОБКА ПРОГРАМНОЇ ЧАСТИНИ ДОДАТКУ

3.1. Огляд реалізації рівня бази даних

В попередньому розділі цієї роботи було вказано, що додаток для екомоніторингу довкілля, що розробляється, складається з двох частин, а саме клієнтської і серверної. В цьому розділі буде розглянуто загальну структуру та реалізацію кожної з частин.

Створений програмний продукт – це мобільний додаток для Android і iOS, що є платформою, яка призначення для реєстрації та контролю випадків порушення екологічних норм, реєстрування природних лих та інших випадків що пов`язані з добробутом навколишнього середовища. Додаток для екомоніторингу довкілля призначений для широкого користувачів. Реалізований додаток дає можливість кожному:

- ознайомитися з уже зареєстованими екологічними проблемами поблизу нього;
- додати екологічну проблему, описавши її та прикріпивши фото і позначку на мапі.

Для реалізації даного продукту було обрано платформу Spring Framework на базі мови програмування Java, як основу для серверної частини додатку і платформу Flutter на основі мови прогрумування Dart.

В реалізованому додатку було використано PostgreSQL, в якості основної БД для збереження даних про користувачів і екологічні інциденти. Розпочнемо з огляду загальної ER(entity-relationship) діаграми зображеної на рис.4.1.

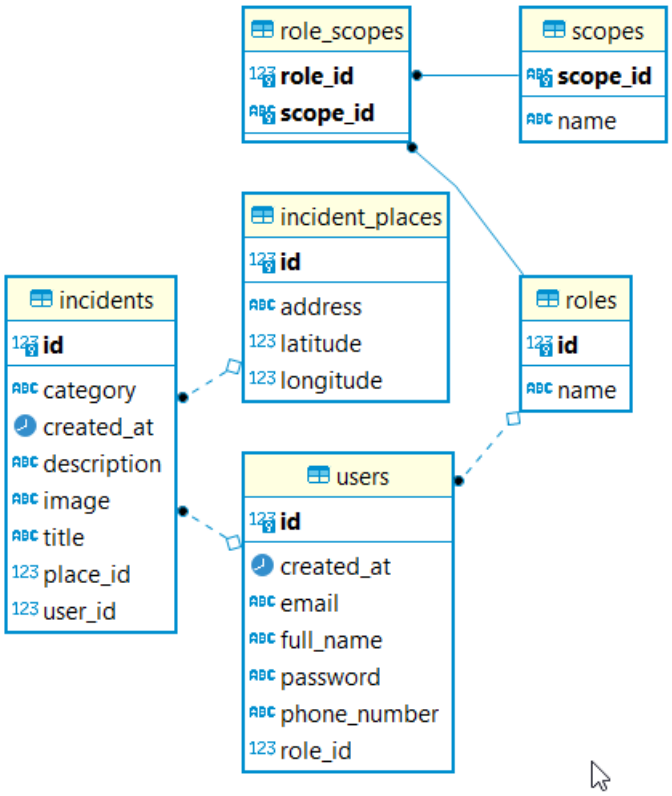

Рис. 3.1. Загальна ER(entity-relationship) діаграма

У наступний таблицях буде детально описано структуру, вміст та відношення з іншими таблицями для кожної з таблиць, що є представленими на діаграмі на рис.3.1.

У табл. 3.1. містяться дані про таблицю users(Користувачі), а саме інформація про поля таблиці, їх тип, опис цільового призначення та обмеження. В ній зберігаються загальні дані про користувача, які можуть бути використані для контанту з ним. Ця таблиця має відношення Many to One з таблицею roles(Ролі) та відношення One to Many з таблицею incidents(Інциденти).

Таблиця 3.1 – Користувачі (users)

Назва	Тип	Опис	Обмеження
Id	INT	Унікальний ідентифікатор користувача	PRIMARY_KEY
Email	VARCHAR(255)	Адреса електронної пошти	UNIQUE_CONSTRAINT
full_name	VARCHAR(255)	Повне ім’я	
Password	VARCHAR(255)	Хеш паролю	
phone_number	VARCHAR(255)	Контактний номер телефону	UNIQUE_CONSTRAINT
created_at	TIMESTAMP	Дата і час створення запису користувача у базі даних	
role_id	INT	Унікальний ідентифікатор ролі у системі	FOREIGN KEY

У табл. 3.2. містяться дані про таблицю roles(Ролі), а саме інформація про поля таблиці, їх тип, опис цільового призначення та обмеження. В ній зберігаються ролі користувача в системі, що будуть заздалегідь записані у SQL таблицю, а саме USER, ADMIN(тобто роль КОРИСТУВАЧ і АДМІНІСТРАТОР). Ця таблиця має відношення Many To Many з таблицею scopes(Дозволи). Для реалізації даного відношення було створено таблицю role_scopes(Дозволи ролей).

Таблиця 3.2 – Ролі (roles)

Назва	Тип	Опис	Обмеження
id	INT	Унікальний ідентифікатор ролі	PRIMARY_KEY
name	VARCHAR(255)	Назва ролі у системі	UNIQUE_CONSTRAINT

У табл. 3.3. містяться дані про таблицю scopes(Дозволи), а саме інформація про поля таблиці, їх тип, опис цільового призначення та обмеження. В ній зберігаються дозволи, які є прикріпленими до ролі в системі, що будуть заздалегідь записані у SQL таблицю. Ця таблиця має відношення Many To Many з таблицею roles(Ролі). Для реалізації даного відношення було створено таблицю role_scopes(Дозволи ролей).

Таблиця 3.3 – Дозволи (scopes)

Назва	Тип	Опис	Обмеження
scope_id	INT	Унікальний ідентифікатор дозволу	PRIMARY_KEY
name	VARCHAR(255)	Назва дозволу у системі	UNIQUE_CONSTRAINT

У табл. 3.4. містяться дані про таблицю role_scopes(Дозволи ролей), а саме інформація про поля таблиці, їх тип, опис цільового призначення та обмеження. Вона містить дані про дозволи, що прикріплені до ролей, а саме пари унікальних ідентифікаторів таблиць roles(Ролі) та scopes(Дозволи). Дозволи є прикріпленими до ролі в системі та будуть заздалегідь записані у SQL таблицю. Ця таблиця була

створена для реалізації відношення Many To Many між таблицями roles(Ролі) та scopes(Дозволи).

Таблиця 3.4 – Дозволи ролей (role_scopes)

Назва	Тип	Опис	Обмеження
scope_id	INT	Унікальний ідентифікатор дозволу	FOREIGN_KEY
role_id	INT	Унікальний ідентифікатор ролі	FOREIGN_KEY

У табл. 3.5. містяться дані про таблицю incidents(Інциденти), а саме інформація про поля таблиці, їх тип, опис цільового призначення та обмеження. В ній зберігаються загальні дані про інціденту, які можуть бути використані для моніторингу і відображення інциденту на графічному інтерфейсі. Ця таблиця має відношення Many to One з таблицею users(Користувачі) та відношення One to One з таблицею incident_places(Місця інцидентів).

Таблиця 3.5 – Інциденти (incidents)

Назва	Тип	Опис	Обмеження
id	INT	Унікальний ідентифікатор інциденту	PRIMARY_KEY
user_id	INT	Унікальний ідентифікатор користувача	FOREIGN_KEY

place_id	INT	Унікальний ідентифікатор місця інциденту	FOREIGN_KEY
category	VARCHAR(255)	Категорія інциденту	
description	VARCHAR(255)	Опис інциденту	
title	VARCHAR(255)	Назва інциденту	
created_at	TIMESTAMP	Дата і час створення запису інциденту у базі даних	
image	VARCHAR(255)	Посилання на картинку	

У табл. 3.6. містяться дані про таблицю incident_places(Місця інцидентів), а саме інформація про поля таблиці, їх тип, опис цільового призначення та обмеження.

Таблиця 3.6 – Місця інцидентів (incident_places)

Назва	Тип	Опис	Обмеження
id	INT	Унікальний ідентифікатор місця інциденту	PRIMARY_KEY
address	VARCHAR(255)	Адреса місця знаходження інциденту	

latitude	FLOAT8	Широта у кординатах місця інциденту	
longitude	FLOAT8	Довгота у кординатах місця інциденту	

В ній зберігаються загальні дані про місце інціденту, які можуть бути використані для моніторингу і відображення інциденту на графічному інтерфейсі. Ця таблиця має відношення One to One з таблицею incidents(Інциденти).

3.2. Огляд реалізації серверної частини

Для реалізації серверної частини нашого додатку було обрано платформу Spring Framework. Серверна частина являє собою 2 сервіси що використовують технологію сервлетів і виконуються на контейнері сервлетів Tomcat [19]. Кінцевим результатом білду кожного з сервісів є файл з розширенням .jar. Це уможливлює просте розгортання обох сервіс без додаткових зусиль. Загалом кожен з них надає API розроблені відповідно до парадигми REST. Впродовж розробки було вирішено поділити весь серверний функціонал на 2 частини, кожна з яких матиме чітку відповідальність [14]. Проведемо огляд функціоналу та призначень даних сервісів.

3.2.1 Огляд monitoring сервісу

Перш за все, варто зазначити, що весь сервіс поділений на каталоги, які містять потрібні класи, в залежності від зони відповідальності.

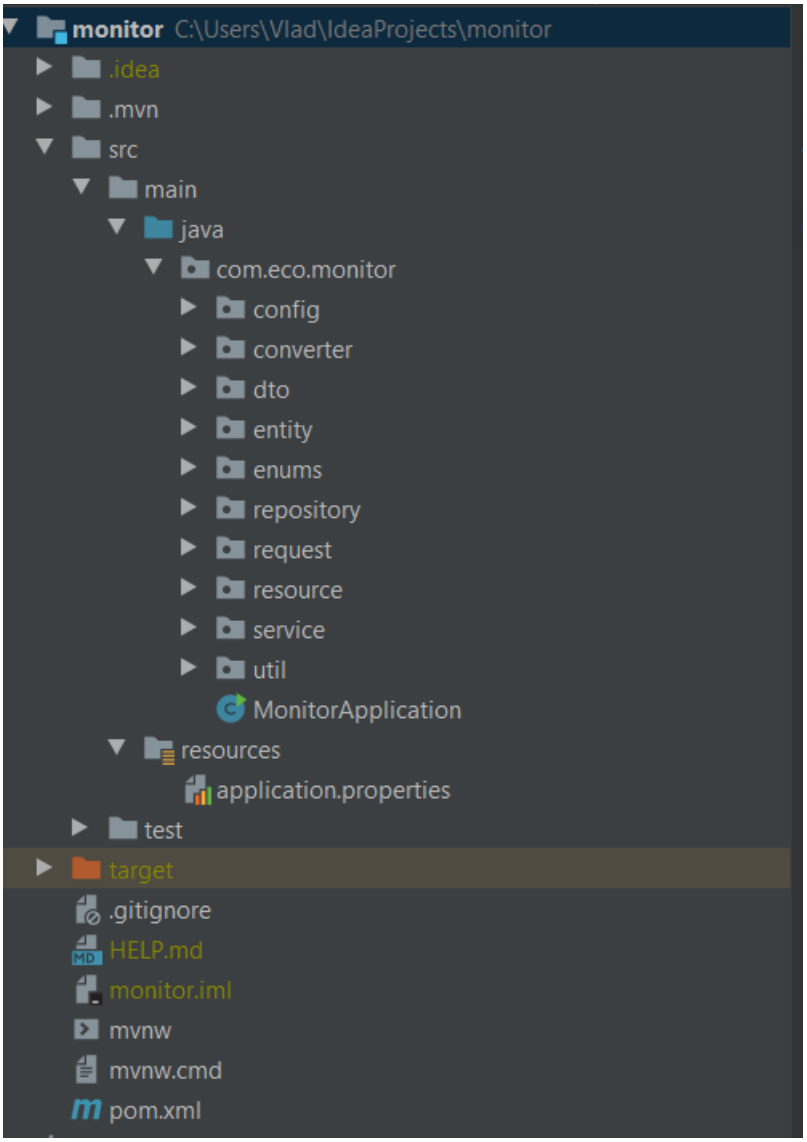

Рис. 3.2. Структура monitoring сервісу

Повний перелік каталогів та структуру monitoring зображено на рис 3.2.

Каталог monitor.config є відповідальним за налаштування усього monitoring сервісу. Він містить ConfigurationManager клас який відповідає за синхронізацію з конфігураційним файлом application.properties, що знаходиться у каталозі resources. За допомогою файлу application.properties ми налаштовуємо цілу низку потрібних речей, таких як:

– корневий шлях для URL додатку
– номер порту

– конфігурація PostgreSQL(url бази даних, драйвер для Java, логін та пароль для адміністратора бази даних)
– секрет для створення або оброблення JWT токенів

ConfigurationManager клас є біном Spring контексту нашого додатку і використовується у інших класах за допомогою механізму Dependency Injection реалізованого Spring Framework [8]. Наприклад, ConfigurationManager використовується у класі що займається генерацією JWT токенів для доступу до секрету, що у свою чергу потрібен для створення або оброблення JWT токенів. Повний перелік класів що входять до поточного каталогу зображено на рис. 3.3.

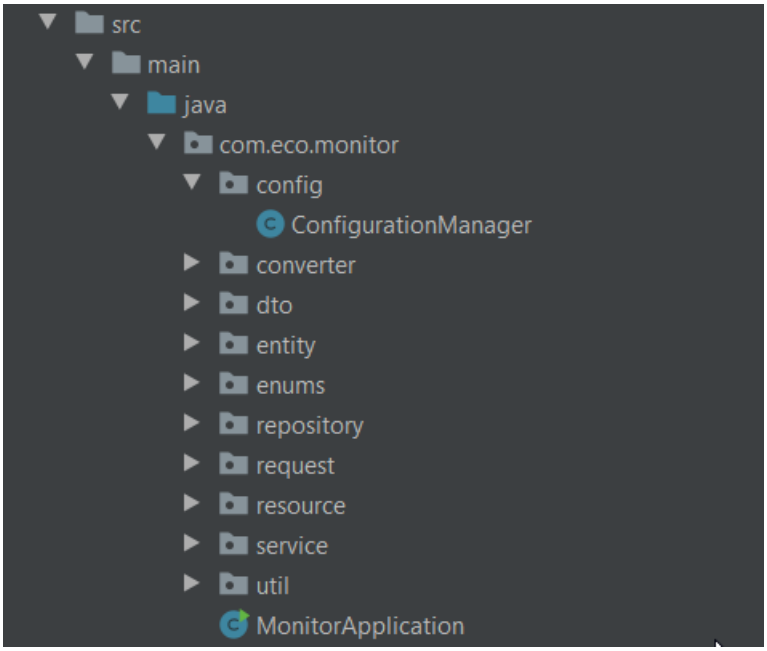

Рис. 3.3. Вміст config каталогу

Розглянемо наступний каталог monitor.converter. Цей каталог призначений для зберігання класів, що будуть відповідальні за перетворення класів одного виду в класу іншого виду. Варто відмітити, що кожна entity у нашому додатку представлена двома класами:

– persistence класи
– DTO класи

Persistence класи – це класи що відповідальні за взаємодію з Hibernate і Spring Data. Вони забезпечують зберігання потрібних для нас сутностей у базі даних, також вони містять назви таблиць, окремих конолок у базі, відношення між сутностями, тощо. Зазвичай опис специфічних для бази даних речей у цих класах відбувається за допомогою Hibernate/JPA анотацій [15].

Абревіатура DTO розшивровується як Data Transfer Object(Об’єкт для траспортування даних). DTO класи використовуються на бізнес і API шарі нашої архітектури. Тобто вони використовуються для видачі даних на користувацький інтерфейс і маніпуляції з даними на рівні сервісів [28].

Повертаючись до призначення каталогу monitor.converter можна сказати, що класи для зберігання яких був створений цей каталог є класи що конвертують або перетворюють persistence класи сутностей в DTO класи сутностей і навпаки.

Повний перелік класів що входять до поточного каталогу зображено на рис. 3.4.

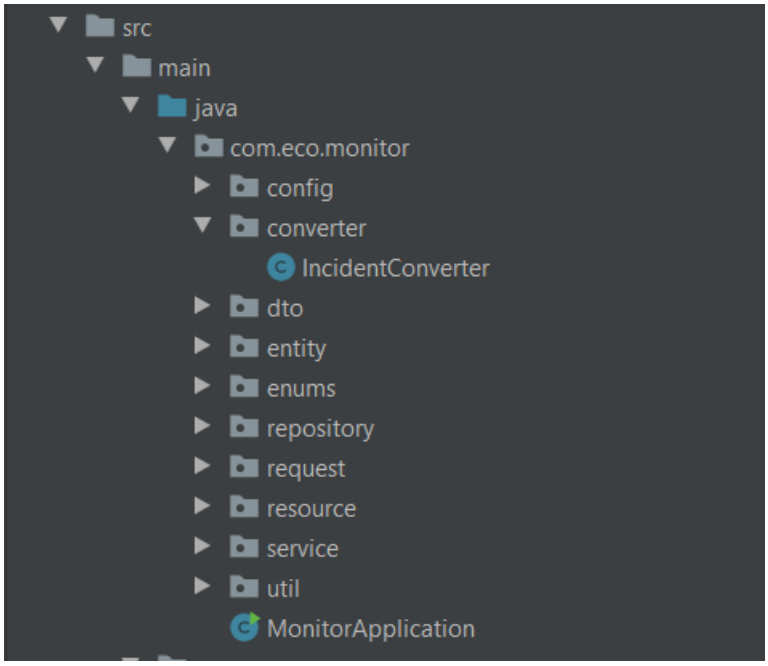

Рис. 3.4. Вміст converter каталогу

Розглянемо наступний каталог monitor.dto. Цей каталог призначений для зберігання DTO класів, що будуть використовуватися на бізнес і API шарі нашої

архітектури. Ці класи є представленням певних сутностей у нашій моделі даних і будуть використовуватися для видачі даних на користувацький інтерфейс і маніпуляції з даними на рівні сервісів.

Повний перелік класів що входять до поточного каталогу зображено на рис. 3.5.

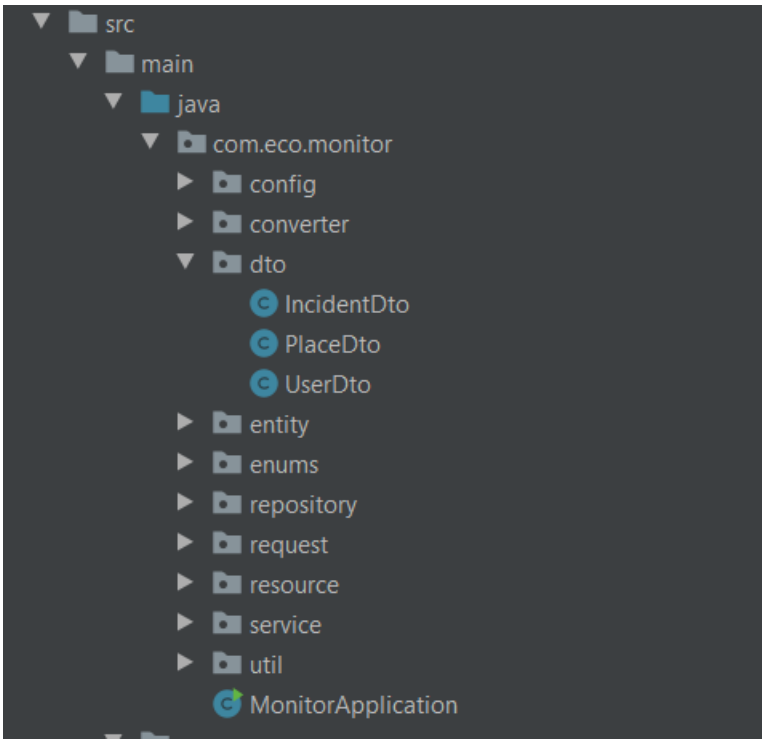

Рис. 3.5. Вміст dto каталогу

Розглянемо наступний каталог monitor.entity. Цей каталог призначений для зберігання persistence класів, що будуть використовуватися на persistence шарі нашої архітектури. Ці класи є представленням сутностей у нашій моделі даних і будуть використовуватися для налаштування взаємодії з базою даних та ORM провайдером.

Суть ORM піходу заключається в тому, що persistence класи у програмному коді співставляються з таблицями у базі даних [20]. Ці класи є гнучким інструментом для налаштування таблиці, колонок, індексів, що виключає безпосереднє написання SQL скріптів для створеннння або налаштування тих чи

інших речей у базі даних. Для усіх конфігурацій використовуються Hibernate/JPA анотації. Відповідно будь-яка зміна чи маніпуляція відповідними Java класами викликає зміни у таблицях бази даних. Використовуючи цей підхід ми значно економимо наш час, який був би витрачений на написання та підтримку SQL скріптів, що займалися б створенням, обновленням або видаленням певних даних з SQL таблиць.

Повний перелік класів що входять до поточного каталогу зображено на рис. 3.6.

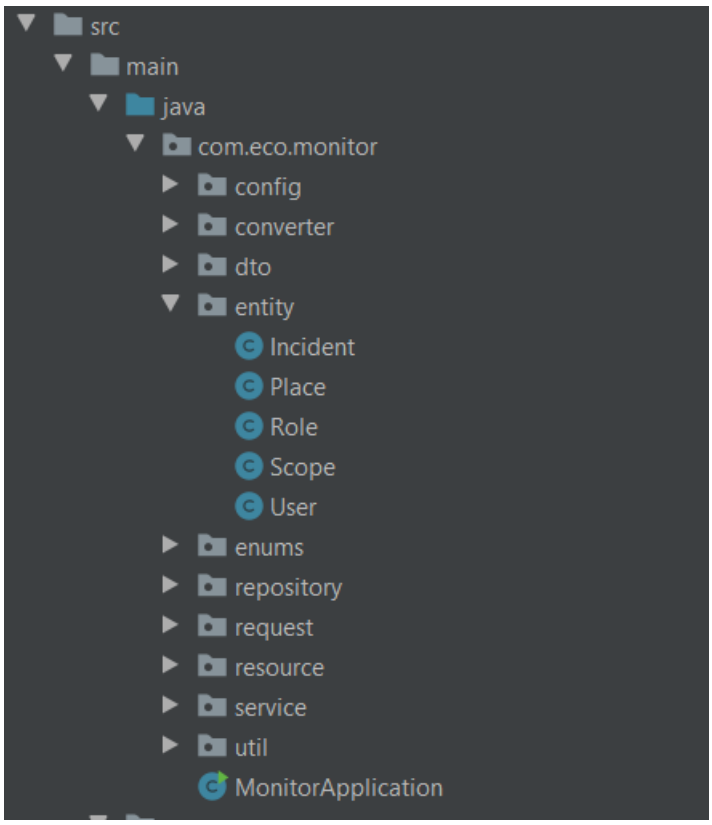

Рис. 3.6. Вміст entity каталогу

Каталог monitor.enums існує для зберігання enum класів або так званих перерахувань, що можуть використовуватися у різних місцях нашого додатку. Це класи, що по суті, створюють окремий тип даних з фіксованим набором значень. До прикладу, наш каталог містить єдиний enum – Role і він має такі значення як USER, ADMIN. Ці значення використовуються для присвоєння ролі користувачу.

Повний перелік класів що входять до поточного каталогу зображено на рис. 3.7.

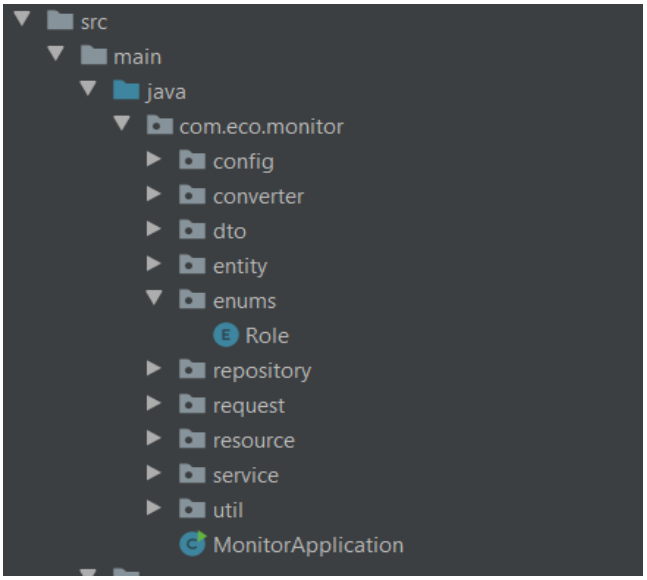

Рис. 3.7. Вміст enums каталогу

Розглянемо наступний каталог monitor.repository. Він містить у собі класи що наслідують інтрейфейс CrudRepository з пакету Spring Data. Метою цієї абстракції Spring є значне зменшення кількості типового коду, необхідного для реалізації доступу до даних, а також доступ до стандратного набору заздалегідь реалізованих методів по взаємодії з бажаної сутністю.

Повний перелік класів що входять до поточного каталогу зображено на рис. 3.8.

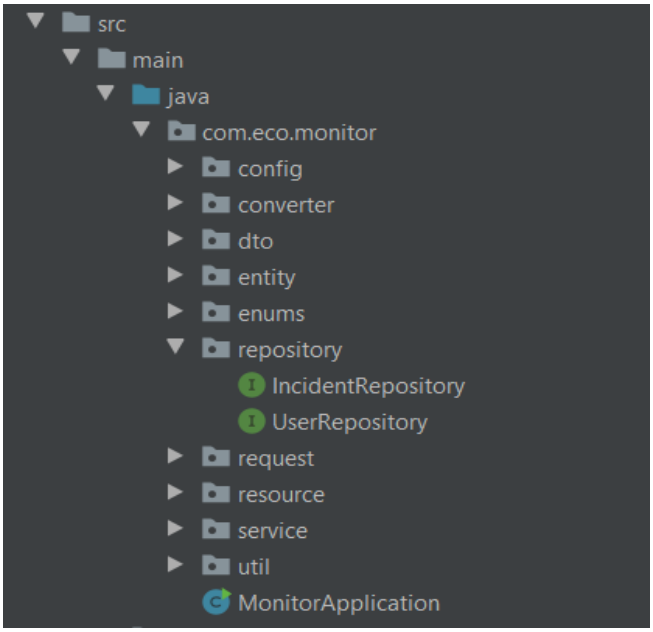

Рис. 3.8. Вміст repository каталогу

Розглянемо наступний каталог monitor.request. Цей каталог призначений для зберігання класів, що будуть використовуватися для обробки даних, що будуть відправлені з клієнта до нашого API. По суті, ці класи являються точним копіями JSON файлів, що ми їх будемо приймати на сервері [18]. Вони використовуються для того щоб json mapper зміг трансформувати дані у Java класи. Ці класи згодом будуть перетворені на DTO класи після того як json mapper успішно перетворить JSON у request клас.

Повний перелік класів що входять до поточного каталогу зображено на рис. 3.9.

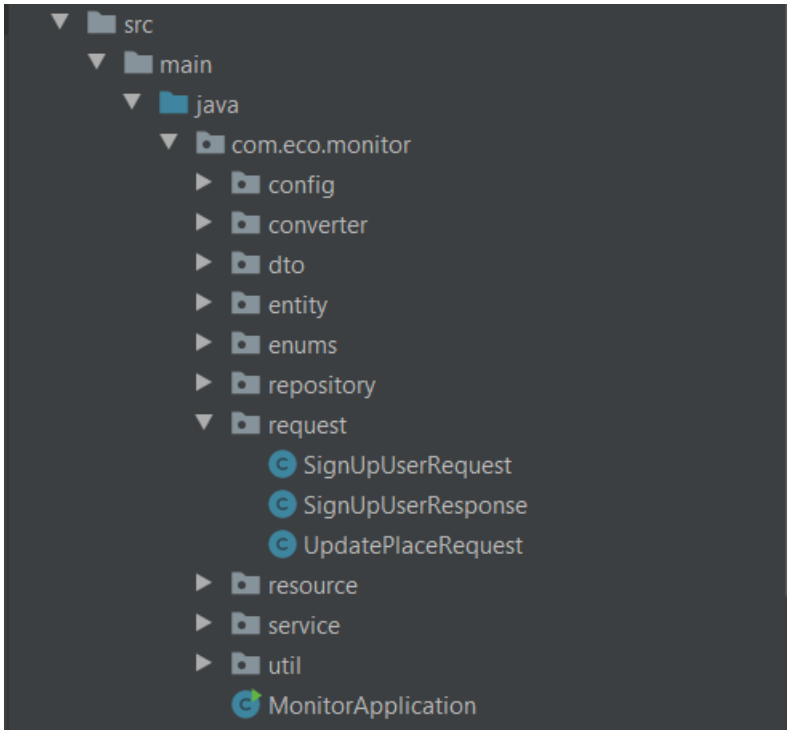

Рис. 3.9. Вміст request каталогу

Одним з найбільш важливих каталогів є – monitor.resource каталог. Класи, що містяться у ньому надають можливість сервісу обробляти HTTP запити, використовуючи при цьому spring-boot-starter-web бібліотеку, яка у свою чергу реалізує сервлети що приймають ці запити за допомогою Tomcat [19]. Це є зручним і простим способом що позбавляє нас від потреби написання і підтримки великої кількості одноманітного коду, що реалізовує сервлети напряму. Ще однією важливою функцією цих класів є перенаправлення даних у шар бізнес логіки. Кожен так званний контролер викликає методи сервісів що реалізують бізнес логіку. Контролер у даному контексті ми називаємо клас, що відповідає за обробку HTTP запиту і подальший виклик методів класів, які реалізують бізнес логіку.

Повний перелік класів що входять до поточного каталогу зображено на рис. 3.10.

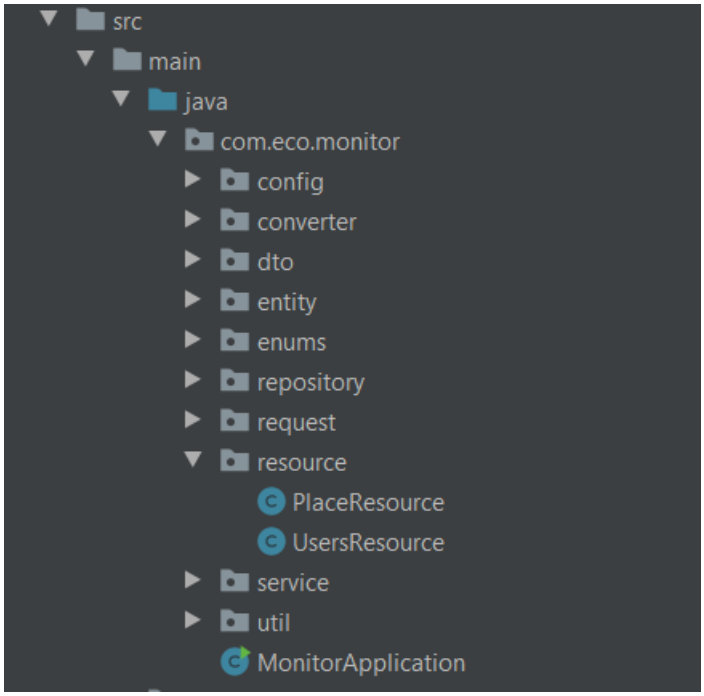

Рис. 3.10. Вміст resource каталогу

А у таблиці 3.7 перераховані можна побачити існуючі контролери, що були реалізовані у поточному каталозі.

Таблиця 3.7 – Контролери

Назва контролера	URL	HTTP метод	Призначення
User Controller	monitoring/users/sign-up	POST	Реєстрація користувача
	monitoring/users/sign-in	POST	Логін користувача
	monitoring/users/{userId}	GET	Отримати користувача
	monitoring/users/{userId}	DELETE	Видалити користувача

	monitoring/users	GET	Отримати всіх користувачів
	monitoring/users/log-out	POST	Логаут користувача
Role Controller	monitoring/roles	GET	Отримати всі ролі
	monitoring/roles/my	GET	Отримати мої ролі
	monitoring/roles	POST	Створити роль
	monitoring/roles/{id}	GET	Отримати роль
	monitoring/roles/{id}	DELETE	Видалити роль
Incident Controller	monitoring/incidents	POST	Створити інцидент
	monitoring/incidents	GET	Отримати всі інциденти
	monitoring/incidents/my	GET	Отримати мої інцидент
	monitoring/incidents/{incidentId}	GET	Отримати інцидент
	monitoring/incidents	PUT	Оновити інцидент
	monitoring/incidents/{incidentId}	DELETE	Видалити інцидент

Розглянемо наступний каталог monitor.service. Цей каталог призначений для зберігання класів, що будуть виконувати бізнес логіку [20]. Зазвичай це сервіс

класи, що виконують якісь перетворення або обчислення і віддають свій результат репозиторіям, які в свою чергу зберігають результат.

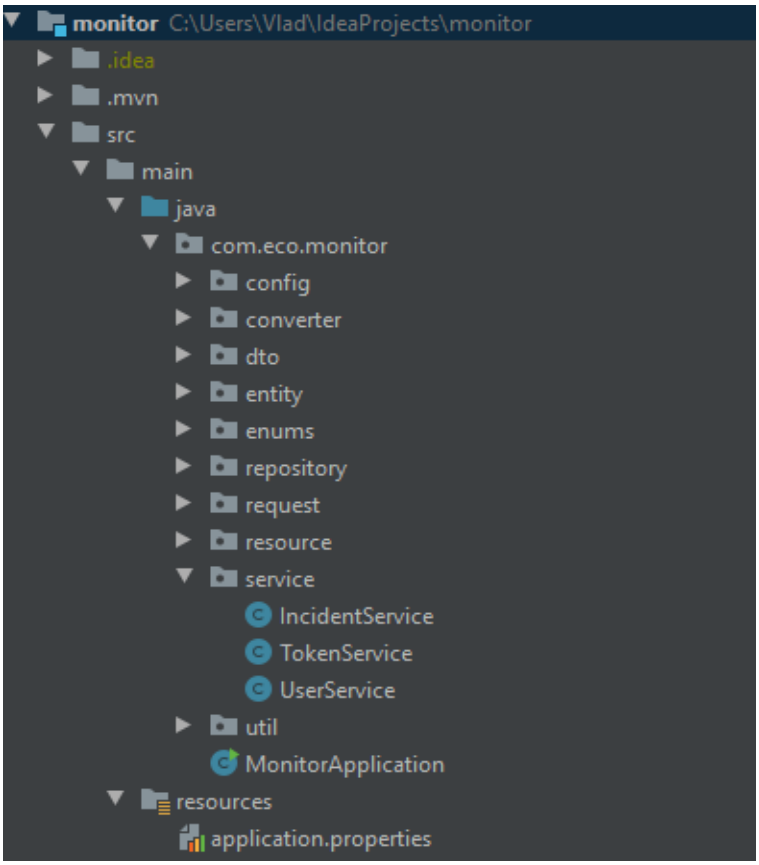

Рис. 3.11. Вміст service каталогу

Повний перелік класів що входять до поточного каталогу зображено на рис. 3.11.

3.2.2 Огляд file сервісу

File сервіс поділений на каталоги, які містять потрібні класи, в залежності від зони відповідальності. Повний перелік каталогів та структуру monitoring зображено на рис 3.12.

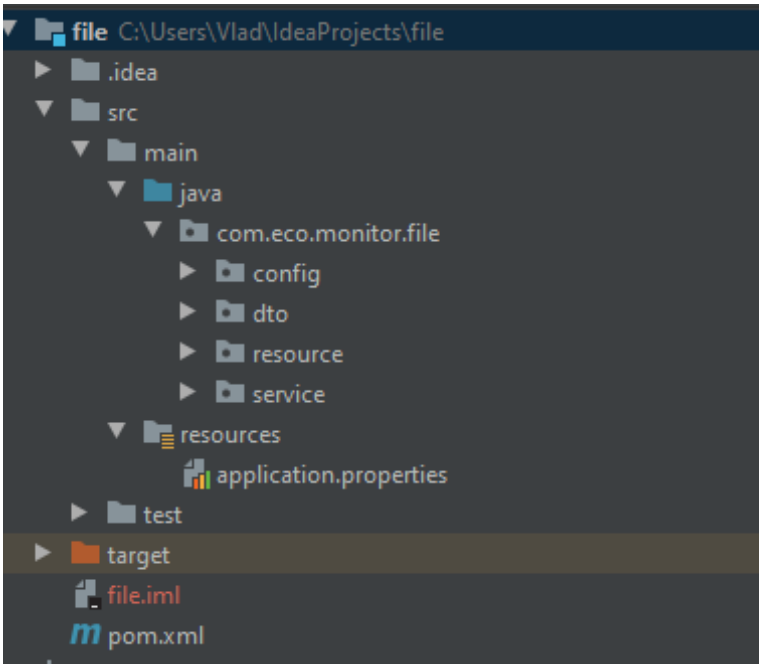

Рис. 3.12. Структура file сервісу

Каталог file.config є відповідальним за налаштування усього monitoring сервісу. Він містить ConfigurationManager клас який відповідає за синхронізацію з конфігураційним файлом application.properties, що знаходиться у каталозі resources. За допомогою файлу application.properties ми налаштовуємо цілу низку потрібних речей, таких як:

– корневий шлях для URL додатку
– номер порту
– конфігурація PostgreSQL(url бази даних, драйвер для Java, логін та пароль для адміністратора бази даних)
– налаштування для AWS S3

ConfigurationManager клас є біном Spring контексту нашого додатку і використовується у інших класах за допомогою механізму Dependency Injection реалізованого Spring Framework. S3Config – це клас, який є відповідальним за налаштування AWS SDK і доступо до S3 бакету, що використовується для зберігання файлів [24].

Повний перелік класів що входять до поточного каталогу зображено на рис. 3.12.

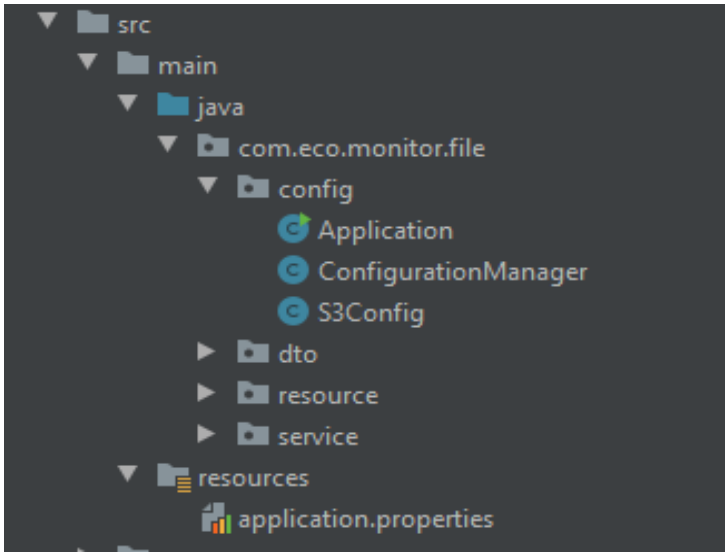

Рис. 3.12. Вміст config каталогу

Розглянемо наступний каталог file.dto. Цей каталог призначений для зберігання DTO класів, що будуть використовуватися на бізнес і API шарі нашої архітектури. Ці класи є представленням певних сутностей у нашій моделі даних і будуть використовуватися для видачі даних на користувацький інтерфейс і маніпуляції з даними на рівні сервісів.

Повний перелік класів що входять до поточного каталогу зображено на рис. 3.13.

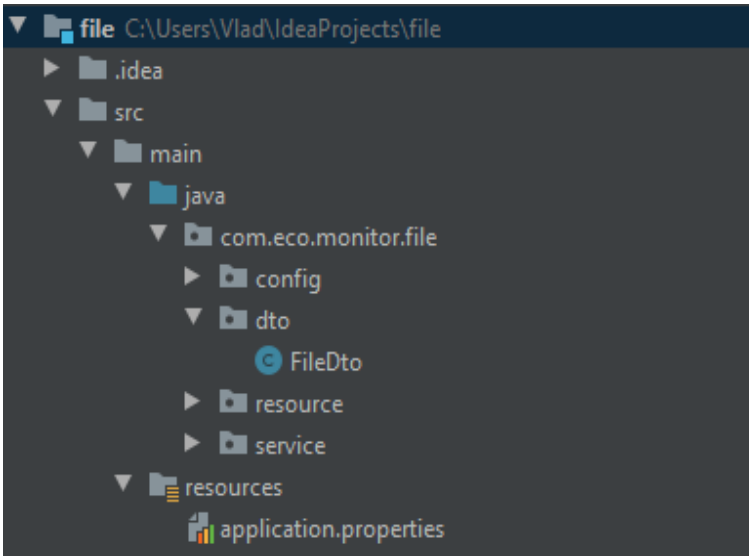

Рис. 3.13. Вміст dto каталогу

По аналогії з monitoring сервісом можемо стверджувати що класи, що містяться у ньому надають можливість сервісу обробляти HTTP запити, використовуючи при цьому spring-boot-starter-web бібліотеку, яка у свою чергу реалізує сервлети що приймають ці запити за допомогою Tomcat.

У цей каталог входить тільки один клас FileResource який відповідає за завантаження і видалення файлів з S3 сховища, що можна побачити на рис. 3.14.

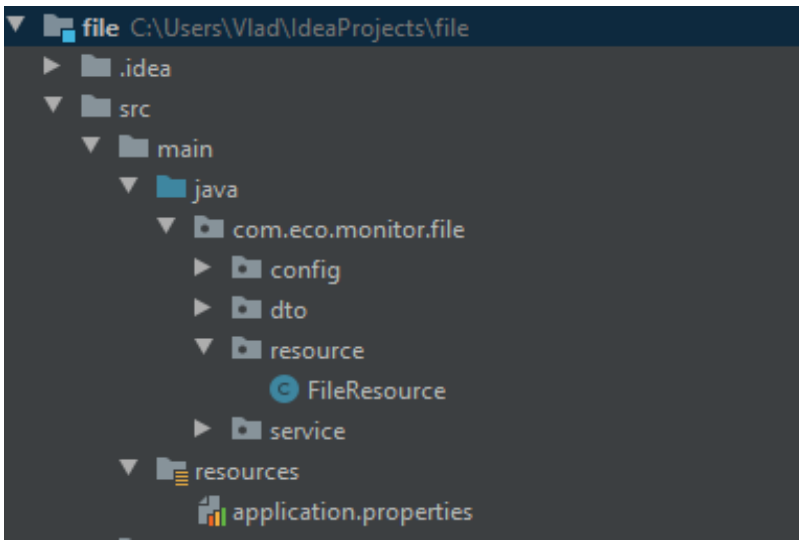

Рис. 3.14. Вміст resource каталогу

По анології з monitoring сервісом, класи що міть file.service каталог відповідають за реалізацію бізнес логіки. Вони займаються перетвореннями,

валідаціями або обчисленнями і віддають свій результат репозиторіям, які в свою чергу зберігають результат. У file сервісі у цьому каталозі маємо 2 класи: S3IntegrationService, S3Service.

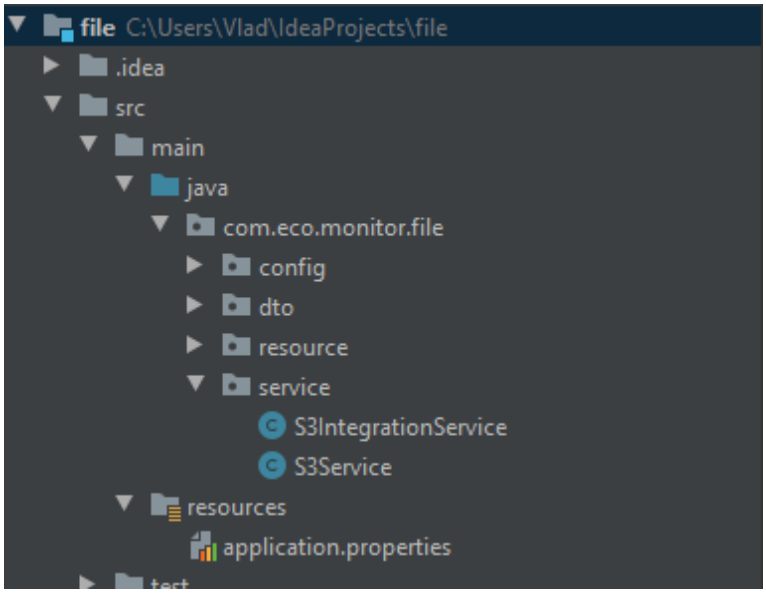

Рис. 3.15. Вміст service каталогу

Перший займається спощенням взаємодії з AWS S3 SDK, по суті, він є ще одним рівнем абстракції. S3Service безпосередньо приймає файл і використовує API щоб пропонує S3IntegrationService для збереження або видалення файлів. Як ці сервіси виглядають у структурі проекту можна побачити на рис. 3.15.

3.3. Огляд реалізації клієнтської частини

Клієнтська частина була реалізована на базі фреймворку Flutter та мови програмування Dart. Основним компонентом мобільного додатку на основі Flutter є віджет. Віджети описують, як повинен виглядати їхній вигляд, враховуючи їх поточну конфігурацію та стан. Для реалізації даного додатку було створено більше велику кількість віджетів що відповідають за переходи, обробку інформації та зв’язок з сервером. Загальна структура Flutter проекту зображена на рис. 3.16.

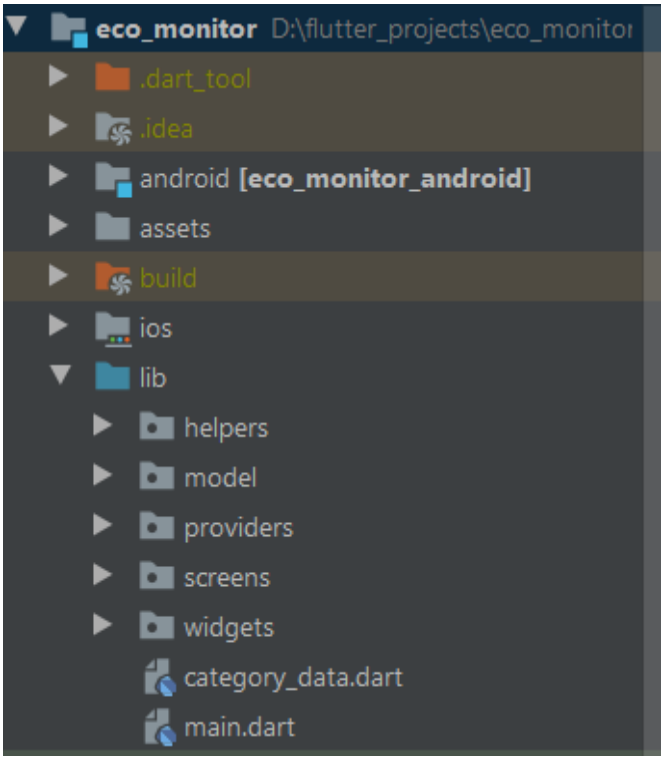

Рис. 3.16. Загальна структура клієнтської частин додатку

Для реалізації спілкування з сервером було використано dio.http бібліотеку, що значно спрощує конструювання HTTP запиту, а також дозволяє зручно та легко обробляти помилки. Ще одна бібліотека, що була використана для взаємодії з сервером це стандартна Dart json бібліотека.

Окремим пунктом варто виділити інтеграцію з сервісами Google Maps. Для реалізації функціоналу даного додатку було використано Maps Static API, Geolocation API, Geocoding API, Places API. Інтеграція була здійснена засобами бібліотеки dio.http взаємодіючи з API напряму без SDK.

Авторизацію реалізовано за допомогою JWT токенів. Логіку для роботи з JWT-токеном реалізовує окремий віджет. Після проходження етапу входу або реєстрації в додаток згенерований токен, що приходить як відповідь на запит зберігається в локальне сховище. При потребі, з нього можна дістати таку корисну інформацію, як ідентифікатор користувача та його роль [23]. При кожному

наступному HTTP-запиті даний токен буде автоматично включений в новий запит, як Authorization хедер.

3.4. Експлуатація

У цьому розділі буде розглянуто основний фунціонал реалізованого додатку для екомоніторингу довкілля. Буде показано і описано головні елементи графічного інтерфейсу.

Запускаючи додаток вперше користувач має 2 опції: вхід або реєстарція.

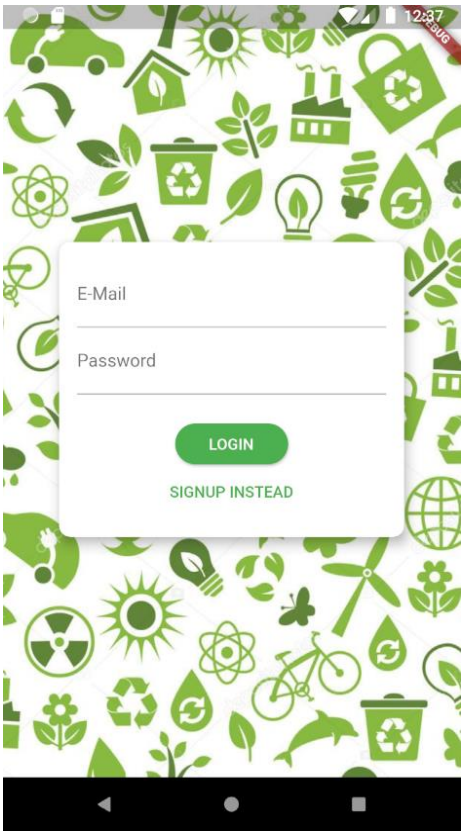

Рис. 3.17. Екран входу в існуючий акаунт

По стандарту користовачу пропонується увійти у свій вже існуючий акаунт (рис. 3.17), але якщо у нього ще немає акаунту він може скористуватися опцією реєстрації натиснувши кнопку “Sign up instead” (рис. 3.18).

Для початку розглянемо екран входу в додаток, що відображеється за замовчуванням при запуску додатку. Для того щоб користувачу увійти в існуючий

акаунт, він повинен заповнити два поля. Для логіну потрібно вказати емейл адресу, до якої прив`язаний вже існуючий акаунт у додаток, а також пароль який був введений під час реєстрації у додатку. Клієнтська частина додатку відсилає HTTP запит на monitoring service, щоб дати змогу користувачеві увійти у систему.

Якщо дані які ввів користувач зходяться з тим що є на сервері, то клієнт отримує відповідь 200 OK і перенаправляє користувача на наступну сторінку.

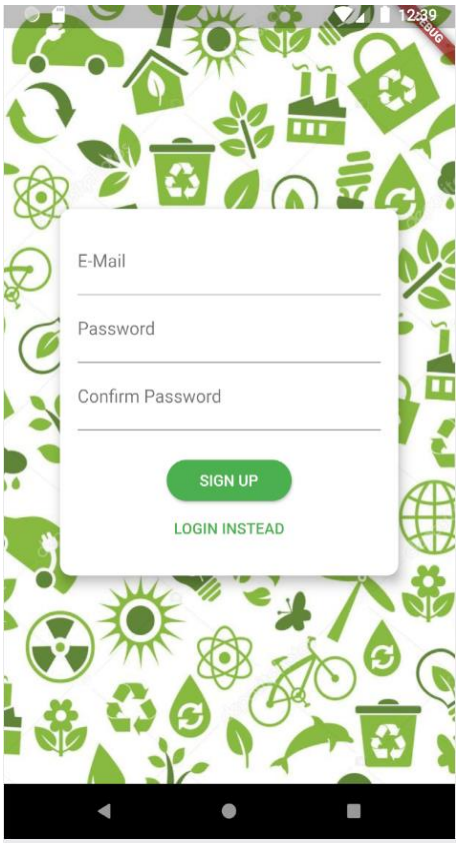

Рис. 3.18. Екран реєстрації нового акаунту

В разі отримання помилики під час валідації клієнт отримує 400 Bad Request відповідь від сервера і відповідне повідомлення про невалідність даних відображається у додатку. У разі успішної відповіді користувач отримує JWT токен, що згодом буде використовуватися для авторизації у наступних запитах до серверу.

Для того щоб зареєструватися у системі користувачі обов`язково заповнити три поля, а саме: емейл, пароль та підтвердити пароль. Варто відмітити, що на

стороні клієнта відбувається валідація формату емейлу, а також перевіряється чи введені паролі були однаковими. Якщо клієнтська валідація пройшла успішно, то додаток відправляє HTTP запит на monitoring service, щоб зареєструвати користувача. Дані проходять повторну вадідацію, а також перевірку на унікальність емейл адреси у системі. Створюється новий запис у табличці, а клієнт отрумує 200 OK відповідь від сервера. В разі отримання помилики під час валідації клієнт отримує 400 Bad Request відповідь від сервера і відповідне повідомлення про невалідність даних відображається у додатку. У разі успішної відповіді користувач отримує JWT токен, що згодом буде використовуватися для авторизації у наступних запитах до серверу.

Увівши коректні дані для реєстрації або входу у вже існучий акаунт користувач буде перенаправлений на стартовий екран додатку(рис. 3.19).

Рис. 3.18. Стартовий екран додатку

Потрапляючи на стартовий екран додатку користувач отримує можливість додати новий інцидент, натиснувши зелену кнопку у правому нижньому куті, або відкрити бургер-меню, що знаходиться у лівому верхньому куті екрану. Почнемо наш огляд з опції додавання нового інциденту. Натиснувши на зелену кнопку зі знаком «+», користувач переходить на новий екран, де у нього є можливість вибрати категорію інциденту(рис. 3.19.).

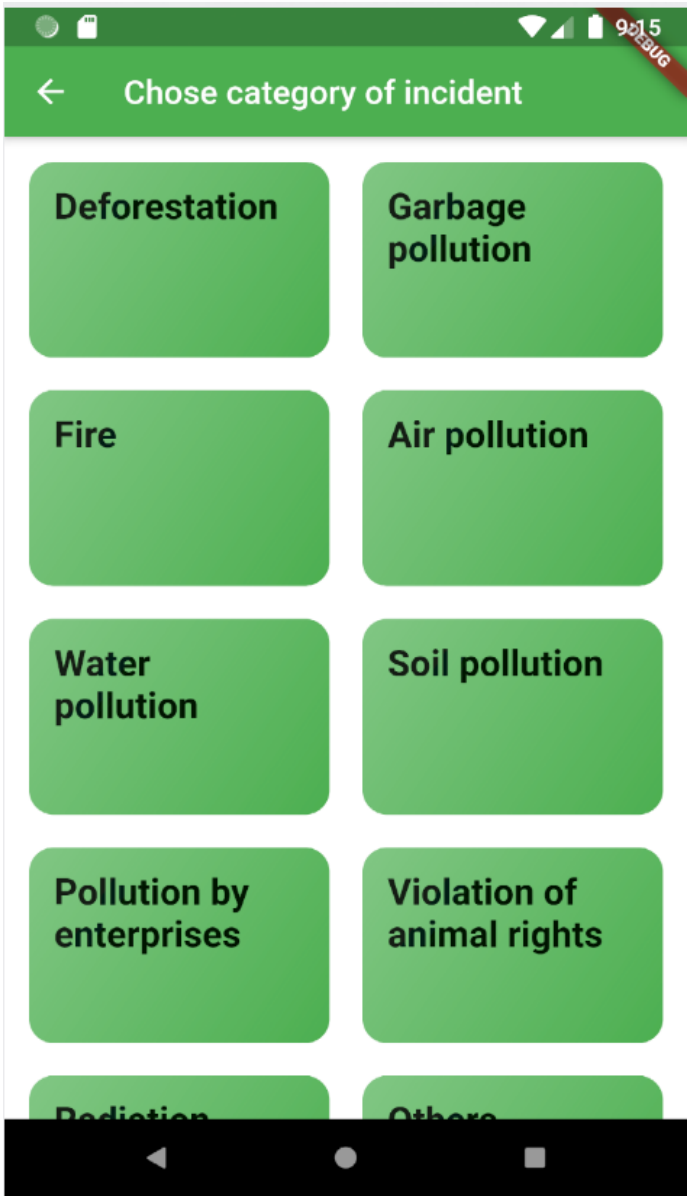

Рис. 3.19. Екран вибору категорії інциденту

На даному екрані користувач може вибрати одну з категорій для екологічного інциденту з запропонованого списку. Якщо його інцидент не підпадає під жодну з існуючих категорі, то користувач може вибрати категорію «Others»(Інше).

Натиснувши на потрібну категорію користувач переходить на новий екран, де у нього є можливість створити новий інцидент(рис. 3.20).

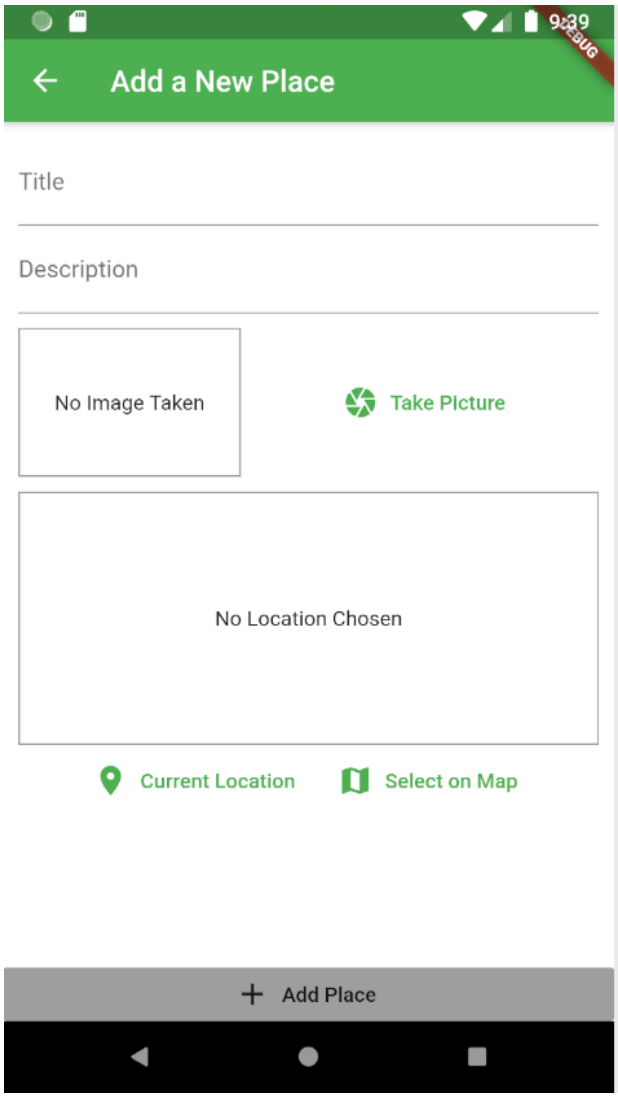

Рис. 3.20. Екран створення інциденту

Щоб створити новий інцидент необхідно заповнити поля «Title» і «Description», а також зробити фотографію з місця інциденту і вибрати локацію.

Щоб зробити фотографію потрібно натиснути «Take Picture» і скориставшись нативної програмою оперативної системи зняти потрібний кадр (рис. 3.21.).

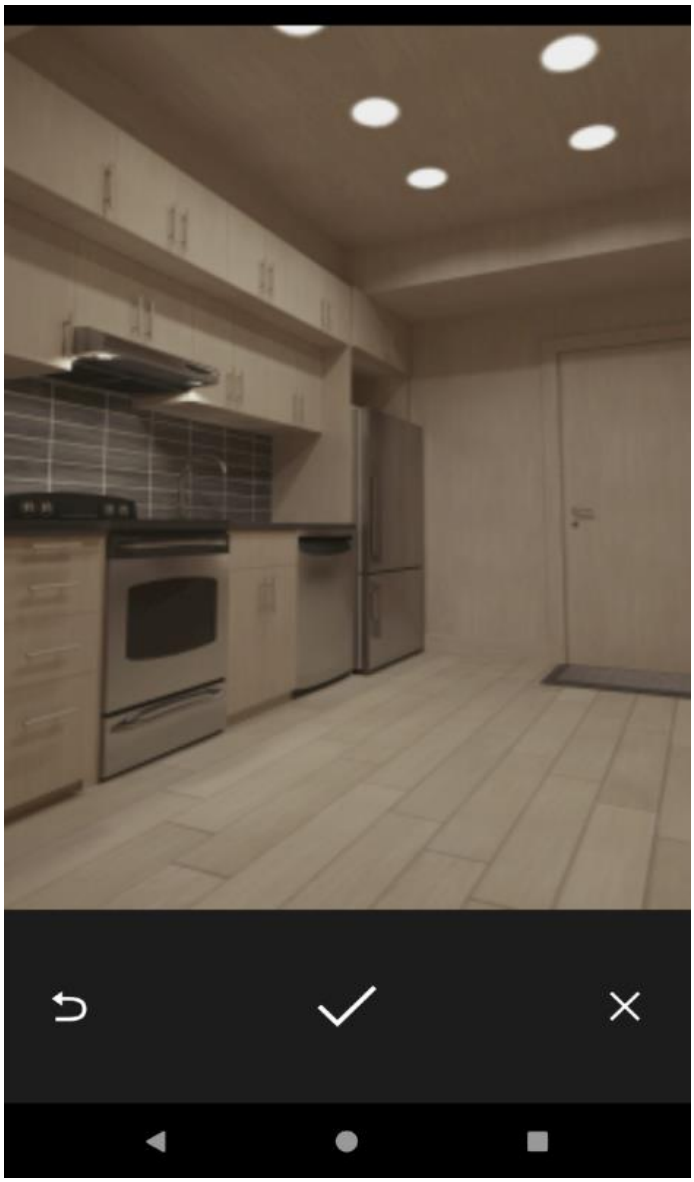

Рис. 3.21. Екран зняття фотографії

Після того як користувач успішно прикріпив фотографію для створення нового інциденту, йому потрібно вибрати локацію цього інциденту. Це можна зробити 2 способами:

– вибрати на карті Google Maps, обравши варіант «Select on Map»

– позначити як локацію своє поточне місце знаходження, обравши варіант «Current Location»

Обравши перший варіант користувач буде перенаправлений на новий екран з мапою від Google Maps, де він з легкістю зможе обрати свою локацію (рис. 3.22).

Рис. 3.22. Екран вибору локації на мапі

Вибравши альтернативний варіани, користувач отримає свої координати залишаючись при цьому на екрані створення нового інциденту (рис. 3.22.).

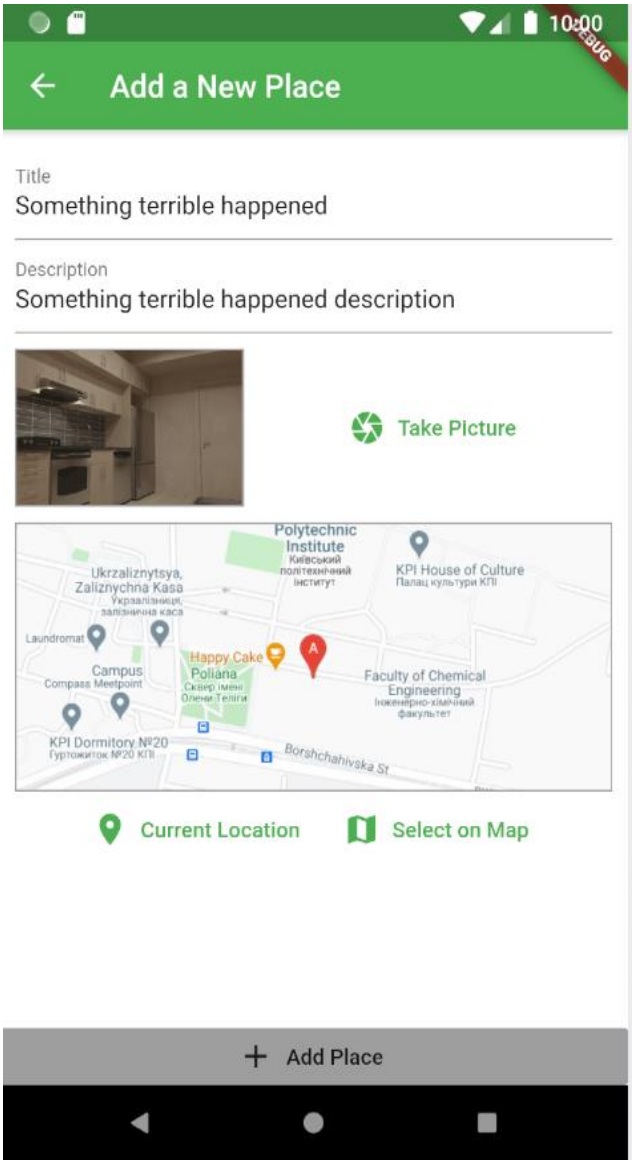

Рис. 3.22. Демонстрація вибору поточної локації

Після заповнення потрібних полів, вибору локації інциденту та потрібної фотографію, користувачу потрібно натиснути на відповідну кнопку внизу екрану щоб додати інцидент у систему. Після натискання цієї кнопки клієнт відправляє фото для завантаження на file сервіс. Після проходження всіх потрібних валідацій фото поміщається на S3 Amazon сховище і тепер воно доступне за посиланням, що спрощує роботу з картинками на клієнті. Отримавши посилання від file сервісу, клієнт направляє усі доступні дані на incident сервіс, щоб додати інцидент у

систему. Створюється новий запис у табличці, а клієнт отрумує 200 OK відповідь від сервера. В разі отримання помилики під час валідації клієнт отримує 400 Bad Request відповідь від сервера і відповідне повідомлення про невалідність даних відображається у додатку.

Успішно створивши інцидент, клієнт повертається на екран з відображенням категорій (рис. 3.19). Там він може створити ще один інцидент, або повернутися на стартовий екран за допомогою стрілки, що розташована у верхньому лівому куті екрану.

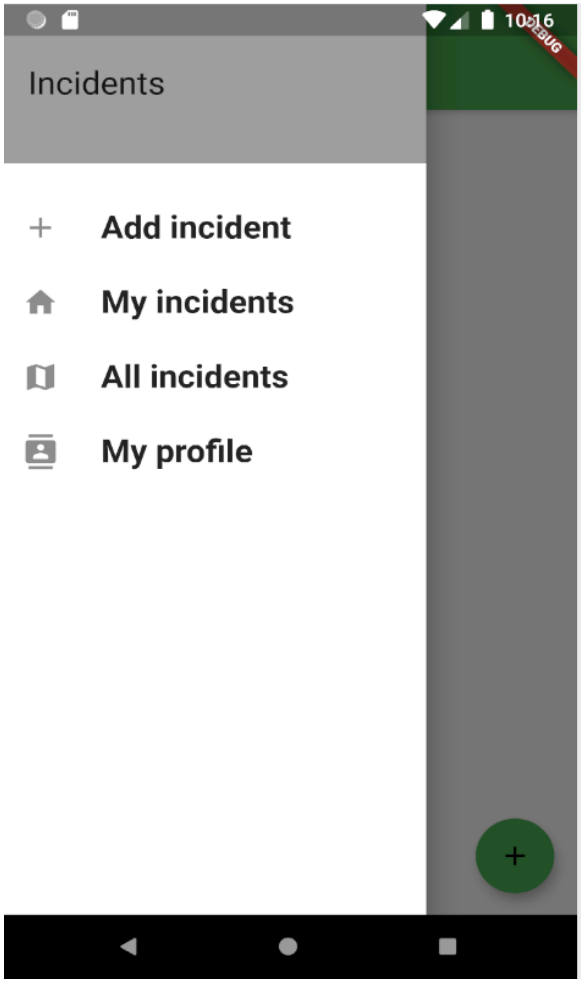

Рис. 3.23. Бургер-меню

Ще одним елементом стартового екрану є бургер меню, що розташоване у лівому верхньому куті екрану. Натиснувши на нього ми побачимо окреме меню, з декількома опціями(рис. 3.23.).

У бургер меню користувач може побачити 4 опції:

- Створення нового інциденту
- Перегляд власних інцидентів
- Перегляд інцидентів інших людей
- Мій профіль

Рис. 3.24. Екран відображення списку власних інцидентів користувача

Кожна з цих опцій веде до нового екрану, який дозволяє користувачу виконати вищезгаді функції. До прикладу, візьмемо екран перегляду своїх інцидентів (рис. 3.24.).

Щоб отримати список інцидентів клієнт робить запит на incident service і отримує список інцидентів по унікальному ідентифікатору користувача. Отримавши відповідь 200 OK, клієнт відображає список інцидентів.

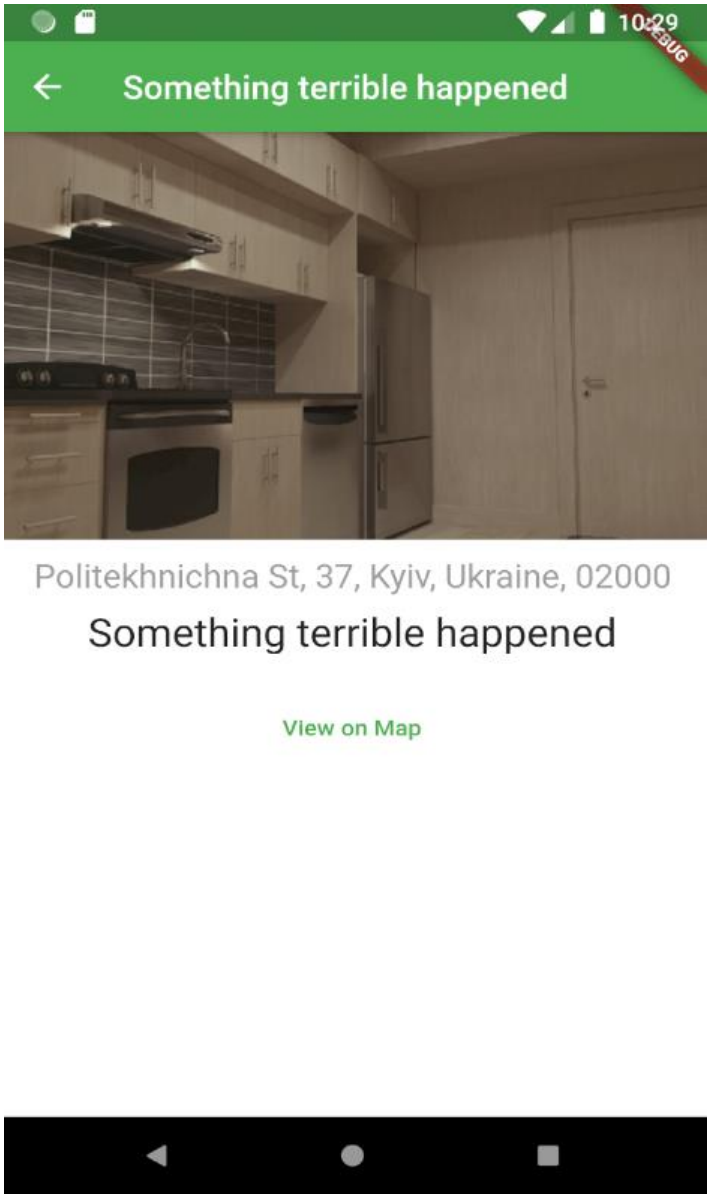

Рис. 3.25. Екран відображення інциденту

Натиснувши на інцидент у списку користувач буде перенаправлений на екран відображення інциденту(рис. 3.25.).

Рис. 3.26. Екран відображення локації інциденту

Для того щоб подивитися на локацію інциденту на мапі, потрібно натиснути кнопку «View on Map», користувач буде перенаправлений на кран відображення локації інциденту(рис. 3.26.).

Висновки до розділу 3

У розділі 3 було проведено детальний огляд реалізації та експлуатації реалізованого додатку, описано підходи і засоби які були використані для імплементації додатку на рівні БД, а також імплементації.

У огляді реалізації рівня БД було представлено ER діаграму для всіх таблиць, а також описано структуру, колонки і зв’язки між ними. Для імплементації рівня БД було використано засоби СУБД PostgreSQL. Для зв’язку між сервером та БД використовувалися такі ORM інструменти як Hibernate і Spring Data.

У огляді реалізації серверної частини було детально описано рішення, що були прийняти для реалізації monitoring і file сервісів. Було розглянуто загальну структуру сервісів, роль, і вміст їхніх каталогів. Було детально розглянуто реалізований API обох сервісів. Серверна частина була імплементована за допомогою Spring Framework і Java.

У огляді реалізації клієнтської частини було розглянуто загальну структуру проекту що був реалізованих засобами фреймворку Flutter та мови програмування Dart. Було детально описано взаємодію клієнтської частини додатку з сервером, а також розглянуто взаємодію з Google Maps.

Також було розглянуто експлуатацію додатку, що є предметом розробки магістерської дисертації. Було детально оглянуто екрани додатку, які є доступними для користувача та описано дії, які може виконувати користувач на даних екранах.

РОЗДІЛ 4

РОЗРОБКА СТАРТАП ПРОЕКТУ

4.1. Опис ідеї проекту

Даний проект у сфері екомоніторингу довкілля є доволі інноватівною ідеєю і не має прямих конкрентів на ринку мобільних додадтків. Розроблений програмний продукт призначений для демонстрації ідеї створення загальної платформи для обговорення проблем, що пов'язані з довкіллям. Люди мають змогу отримати прозорий моніторинг щодо того що відбувається на теренах їньої держави, регіону з точки зору екології. Дані зібрані реалізованим додатком можуть також бути використані державою або приватними компаніями, яких цікавлять досліждення екологічної ситуації. Платформа також може служити доволі ефективний майданчиком для рекламодавців, що зацікавлені у аудиторії, яку цікавить захист нашої планети. Згодом, можна спланувати чіткий план щодо додавання додаткового функціоналу, а також скласти стратегію дій для розробки ефективного і корисного для суспільства додатку [28].

Отже, в ході реалізації дипломного проекту, було реалізовано кроссплатформений мобільний додаток для екомоніторингу довкілля. Даний розділ буде містити стартап-проекту данного додатку. В таблиці 4.1 буде описано вміст ідеї, напрямки застосування та вказано вигоди для користувачів.

Таблиця 4.1.

Опис ідеї стартап-проекту

Зміст ідеї	Напрямки	Вигоди для користувача
Мобільний додаток для екомоніторингу довкілля	Створення екологічних інцидентів	Користувач має змогу зафіксувати, оцифрувати і зробити певну

		екологічну проблему публічною.
	Можливість переглядання екологічних інцидентів у своєму регіоні	Користувач має змогу моніторингу власного регіону на предмет порушень у сфері екологічного добробуту
	Можливість пошуку екологічних інцидентів	Користувач може знаходити екологічні інциденти за категорією і локацією

Зробимо аналіз потенційних техніко-економічних переваг ідеї та чим вона відрізняється від існуючих аналогів та замінників. Конкурентами даного рішення є такі ресурси як CEHOC, Каратєль, Eyewitness та Trashout.

Таблиця 4.2.

Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№	Техніко-економічні характеристики	Мій проект	Конкуренти				W	N	S
			1	2	3	4			
1	Доступність на обох платформах Android і iOS	+	-	+	-	+			+
2	Можливість моніторингу	+	+	-	-	+		+	

	екологічних проблем								
3	Можливість оперування мапою і збереження локації інциденту на мапі	+	+	-	-	-			+
4	Можливість комунікації у коментарях	-	-	+	-	+	+		

Вказаний перелік слабких, сильних та нейтральних сторін ідеї даного продукту дає можливість йому конкурувати на ринку уже існуючих товарів. З табл. 4.2 слідує, що основними перевагами продукту є доступність на обох платформах Android і iOS і можливість оперування мапою все у додатку. На даний момент, недоліками додатку є те, що у ньому немає вбудованої можливості комінкувати у коментарях під інцидентом.

4.2. Технологічний аудит проекту

Таблиця 4.3.

Технологічна здійсненність ідеї проекту

№	Ідея проекту	Технології її реалізації	Наявність технології	Доступність технологій
1	Мобільний додаток для екомоніторингу довкілля	Мова програмування Java	Є в наявності	Доступні безкоштовно

2		Мова програмування Dart	Є в наявності	Доступні безкоштовно
3		База даних PostgreSQL	Є в наявності	Доступні безкоштовно
4		Фреймворк Spring Framework	Є в наявності	Доступні безкоштовно
5		Фреймворк Flutter	Є в наявності	Доступні безкоштовно

У ході планування розробки предмету дослідження магістерської дисертації було вирішено розробити мобільний додаток з використанням клієнт-серверної архітектури. Для розробки серверної частини використано технології Java, Spring Framework, PostgreSQL. Для клієнтської частини було використано мову програмування Dart і фреймворк Flutter.

4.3. Аналіз ринкових можливостей запуску стартап-проекту

Проведемо аналіз ринкових можливостей запуску стартап проекту. В таблиці 4.4. наведено попередню характеристику потенційного ринку стартап-проекту.

Таблиця 4.4.

Попередня характеристика потенційного ринку стартап-проекту

№	Показники стану ринку	Характеристика
1	Кількість головних гравців, од	<5
2	Динаміка ринку (якісна оцінка)	Сфера екологічного моніторингу перебуває у стагнації

3	Наявність обмежень для входу (вказати характер обмежень)	Необхідність залучення інвестицій для підняття авторитетності додатку, як платформи для вирішення екологічних проблем
4	Специфічні вимоги до стандартизації та сертифікації	-

Таблиця 4.5.

Характеристика потенційних клієнтів стартап-проекту

№	Потреба, що формує ринок	Цільова аудиторія	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Необхідність надавання екологічним проблемам публічності	Користувачі, які хочуть вирішувати екологічні проблеми у своєму регіоні	-	Наявність хорошого інтернет сигналу та смартфону з камерою
2	Необхідність моніторингу екологічного стану довкілля	Корпоративні клієнти, дослідники, науковці, активісти, звичайні користувачі	Різниця у масштабі моніторингу та форматі даних які будуть моніторитися	-

Таблиця 4.6.

Фактори загроз

№	Фактор	Зміст загрози	Можлива реакція компанії
1	Недостаній авторитет додатку, як платформи для вирішення екологічних проблем	Недовіра людей до додатку, як інструменту впливу на суспільству і способу змінити екологічну ситуацію	Побудова чіткого маркетингово плану і його імплементація з першого дня запуску, співпраця з державними інституаціями
2	Економічний	На перших етапах розвитку продукту невелика аудиторія не великих можливостей для рекламодавців, що будуть розгляти додаток	Побудова чіткого бізнес-плану із залученням інвесторів

Таблиця 4.7.

Фактори можливостей

№	Фактор	Зміст можливості	Можлива реакція компанії
1	Приваблення інвесторів	Створення плану що задовільнить інвесторів, привабить додаткові інвестиційні ресурси	Збільшити штат, прискорити вихід на ринок, вийти на період самоокупності раніше
2	Співпраця з державними та міжнародними інституціями	За допомогою співпраці з державними і міжнародними інституціями ми зможемо не тільки отримати	Збільшувати штат, розгорнути спланована стратегічну PR кампанію

		додаткові грошові ресурси, а й збільшити довіру до додатку, як платформи для вирішення екологічних проблем	

Таблиця 4.8.

Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентос-проможною)
Новий ринок	Ринок є новим в даній галузі, прямих конкуретнів у сфері немає, але наявні конкуренти з подібною функціональністю	Провести якісну рекламну кампанію для зайняття лідируючих позицій на ринку
Глобальний ринок	Метою додатку є робота на світовому ринку	Співпраця з місцевими державними інституціями і підприємствами

Таблиця 4.9.

Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти в галузі	Клієнти	Товари-замінники
	-	Каратєль, TrashOut, EyeWitness,	Звичайні користувачі, дослідники,	Open Source-рішення

		СЕМОС	науковці, приватні компанії	
Висновки:	У зв’язку з тим що прямих конкуренті у галузі немає необхідно займатися систематичн им розвитком продукту	Є маса додатків, що зможуть виконувати такі ж операції, якщо вони трохи змінять напрям свого розвитку	Клієнти не мають аналогічних продуктів на ринку, тому треба забезпечити максимальну взаємодію з ними	Поки що немає безкоштовних аналогів нашого додатку

Таблиця 4.10.

Обґрунтування факторів конкурентоспроможності

№	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Першість на ринку	Так як проект немає прямих аналогів на ринку, є доволі непогані шанси використати лідируючу позицію для закрпілення у галузі
2	Кросплатформеність	Переважна більшість створюють додаток під конкретну мобільну платформу. Розробка додатку що доступний на багатьох платформах залучить потенційно більше користувачів
3	Можливість роботи з	Більшість непрямих конкурентів не

	інтерактивною мапою	мають можливість роботи з інтерактивною мапою

Таблиця 4.11.

Порівняльний аналіз сильних та слабких сторін

№	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з розроблюваним продуктом						
			-3	-2	-1	0	1	2	3
1	Першість на ринку	19	✓						
2	Кросплатформеність	12			✓				
3	Можливість роботи з інтерактивною мапою	10				✓			

Таблиця 4.12.

SWOT- аналіз стартап-проекту

Сильні сторони: Кросплатформеність Першість на ринку Зручний інтерфейс Можливість роботи з інтерактивною мапою	**Слабкі сторони:** Відсутність ринку, а отже будь-якого досвіду у галузі Відсутність довіри до додатку, як платформи для вирішення екологічних проблем
Можливості: Приваблення інвестицій Монетизація додатку за рахунок реклами та продажу даних	**Загрози:** Недовіра від державних інституцій Неможливість здобуття довіри до додатку, як платформи для вирішення екологічних проблем

Таблиця 4.13.

Альтернативи ринкового впровадження стартап-проекту

№	Альтернатива ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Робота проекту орієнтована на виключно на державу	Висока, оскільки державам потрібні дані про екологічну ситуацію у державі	Залежить від конкретних державних інституцій, від кількох місяців до року
2	Участь у фандрейзингу	Середня, так як сьогодні дуже багато людей, що піклуються про стан довкілля, але в свою чергу не всі готові вносити пожертви	Декілька місяців

4.4. Розроблення ринкової стратегії проекту

Таблиця 4.14.

Вибір цільових груп потенційних споживачів

№	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Небайдужі до екології користувачі	Готові сприйняти негайно	Продукт необхідний	Відсутня	Вхід помірно-складний
2	Приватні компанії	Не всі користувачі готові прийняти	Продукт необхідний не всім	Конкуренція середня	Вхід помірно-складний

3	Державні інституції	Не всі користувачі готові прийняти продукт	Частоково зацікавлені в продукті	Конкуренція середня	Вхід помірно-складний
Які цільові групи обрано: Потрібно працювати зі всіма цільовими групами, перевагу небайдужим до екології користувачам					

Таблиця 4.15.

Визначення базової стратегії розвитку

№	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентос-проможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
1	Розвиток продукту шляхом здобуття довіри авторитетності додатку як платформи для вирішення екологічних проблем, монетизація за рахунок реклами і продажу даних	Впливати на обрані цільові групи	Залучення міжнародних екологічних організацій до співпраці, реклама	Відповідати на побажання користувачів та інвесторів

Таблиця 4.16.

Визначення базової стратегії конкурентної поведінки

№	Чи є проект «пер-шопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
1	Так, проект є «перше-прохідцем»	Так як, проект є «перше-прохідцем» то усі сили будуть направлені на пошук нових споживачів	Так як, проект є «перше-прохідцем» то усі сили будуть направлені на розроблення автентичних характеристик продукту	У разі виникнення конкурентів забезпечити більш ефективний і зручний спосіб роботи з додатком

Таблиця 4.17.

Визначення стратегії позиціонування

№	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкуренто-спроможні позиції власного стартап- проекту	Вибір асоціацій, які мають сфо-рмувати комплексну позицію власного проекту (три ключових)
1	Кросплатфор-меність, можливість	Відповідати на побажання користувачів та	Першість на ринку, кросплатфор-	Екомоніториг, екологічний додаток, робота

	роботи з інтерактивною мапою	інвесторів	меність, можливість роботи з інтерактивною мапою	з захистом довкілля

4.5. Розроблення маркетингової програми стартап-проекту

Таблиця 4.18.

Визначення ключових переваг концепції потенційного товару

№	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити
1	Моніторинг екологічної ситуації у своєму регіоні	Ефективний і зручний додаток, доступний на основних мобільних платформах, що полегшує можливість моніторити екологічну ситуацію	Кросплатформеність, зручний інтерфейс, можливість роботи з інтерактивною мапою
2	Можливість роботи з інтерактивною мапою при звітуванні про екологічні проблеми	Зручний інструмент для відстеження локацій екологічних інцидентів	Відсутність спеціалізованих екологічних додатків, що використовують інтерактивну мапу

Таблиця 4.19.

Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові

<table>
<tr><td></td><td colspan="2"></td></tr>
<tr><td>I. Товар за задумом</td><td colspan="2">Мобільний додаток для екомоніторингу довкілля</td></tr>
<tr><td rowspan="5">II. Товар у реальному виконанні</td><td>Властивості/характеристики</td><td>Розмір</td></tr>
<tr><td>1. Мобільний додаток на базі фреймворку Flutter</td><td>440МБ</td></tr>
<tr><td>2. Сервений додаток на базі платформи Spring Framework</td><td>350МБ</td></tr>
<tr><td colspan="2">Якість: тестування програми на середовищі, логування виключень, перевірка продуктивності роботи додатку.</td></tr>
<tr><td colspan="2">Пакування: код продукту не передбачений для продажу. Доступ до нього буде вільним через AppStore і PlayStore.</td></tr>
<tr><td></td><td colspan="2">Марка: логотип та назва товару.</td></tr>
<tr><td rowspan="2">III. Товар із підкріпленням</td><td colspan="2">До продажу: програмний продукт не призначений для продажу.</td></tr>
<tr><td colspan="2">Після продажу: програмний продукт не призначений для продажу.</td></tr>
</table>

Таблиця 5.20

Визначення меж встановлення ціни

№ п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі становлення ціни на товар/послугу
1	-	-	>2000$	0-20$

Таблиця 4.21.

Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	Доступ до додатку через AppStore і PlayStore.	Хостинг і оновлення додатку на AppStore і PlayStore.	Одноріневий	Вертикальна

Таблиця 4.22.

Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Через онлайн-ресурси та рекламу дізнаються про продукт	Інтернет зустрічі, конференції	Мобільний додаток для екомоніторингу довкілля	Привернути увагу потенційних користувачів до додатку, підняти важливість вирішення екологічних проблем	Показати вплив додатку на вирішення екологічних проблем

Висновки до розділу 4

У четвертому розділі було описано стартап-проекту для розроблюваного програмного продукту, здійснено огляд ідеї проекту, проведено детальний аналіз переваг і недоліків програмного продукту, зроблено огляд можливостей та конкурентів, що присутні на ринку додатків призначених для екомоніторингу довкілля. Для розробки стартап-проекту було здійснено:

- Зроблено детальний огляд ідеї проекту, виділено основні переваги і недоліки програмного продукту, описаний функціонал продукту, досліджено ринок додатків призначених для екомоніторингу довкілля, визначено основних конкрентів та розроблено детальну стратегію виходу на ринок .
- Зроблено технологічний аудит, проведено огляд використаних технологія для розробки програмного продукту, визначено можливості реалізації і платформи на яких буде доступний даний програмний продукт.
- Проаналізувавши ринкову ситуацію, було розблено план, ринкову стратегію для розвитку програмного продукту. Було порівняно перспективи розвитку конкурентів з перспективами розвитку розробленого програмного продукту, визначено цільові групи та розроблено стратегію просування продукту.

В результаті, було розроблено стартап-проект для запуску реалізованого програмного продукту на ринок, набуто знання щодо створення стартап-проектів, дослідження переваг і недоліків продукуту, аналізу та досліження ринку продукту.

ВИСНОВКИ

Ціллю магістерської роботи була розробка мобільного додатку для екомоніторингу довкілля. У результаті виконання роботи була досягнення її мета – розробка гнучкого, зручного додатку, що буде слугувати інструментом для вирішення екологічних питань пов’язаних із захистом довкілля.

Було здійсного огляд та аналіз області дослідження, а також розглянуті аналоги розробленого програмного продукту. Для досягнення мети було імплементовано мобільний додаток з використанням клієнт-серверної архітектури, який можна умовно розділити на 3 рівні:

- рівень БД
- серверна частина
- клієнтська частина

Було проведено обширний аналіз засобів і технологій, що можуть бути використані для реалізації даного програмного продуктую Плануючи, було вирішено використати в якості бази даних – PostgreSQL, для реалізації серверної частини було вирішено використати Java і фреймворк Spring Framework, для реалізації клієнтської частини було обрано мову програмування Dart та фреймворк Flutter. Для збереження файлів було використано сховище Amazon S3, а для реалізації функціоналу інтерактивної мапи було використано Google Maps.

У ході виконання роботи було реалізовано програмний продукт, що дозволяє здійснювати екомоніторинг довкілля і додати екологічні інциденти до системи власноруч. Було реалізовано функціонал, що дозволяє прикріпляти фотографії до інциденту. Також була імплементована можливість обирання локації на інтерактивній мапі.

Також було розоблено стартап-проект для даного програмного продукту. Було визначено переваги та недоліки продукту, визначено стратегію розвитку продукту на цільовому ринку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Андрієнко М. В. Аналіз і адаптація кращих європейських практик щодо реалізації державної екологічної політики на регіональному рівні / М. В. Андрієнко, В. С. Шако // Інвестиції: практика та досвід. – 2017. – No 19 (жовтень). – С. 51–58.

2. Екологічна ситуація в Україні. [Електронний ресурс] // Osvita.ua – 2015. Режим доступу до ресурсу: https://osvita.ua/vnz/reports/ecology/18874/, Дата доступу до ресурсу: 20.11.2020

3. Вісім екологічних проблем України. [Електронний ресурс] // Finance UA – Режим доступу до ресурсу: https://news.finance.ua/ua/news/-/235280/visim-ekologichnyh-problem-ukrayiny, Дата доступу до ресурсу: 20.11.2020

4. Система екологічного моніторингу оточуючого середовища «СЕМОС» [Електронний ресурс] // SEMOS. – Режим доступу до ресурсу: http://ligaoao.ru/eco/semos, Дата доступу до ресурсу: 20.11.2020

5. Всеукраїнська книга скарг «Каратєль». [Електронний ресурс] // Karatel – Режим доступу до ресурсу: https://karatel.foundation101.org/, Дата доступу до ресурсу: 20.11.2020

6. Eyewitness. [Електронний ресурс] // Eyewitness Global. – Режим доступу до ресурсу: https://eyewitness.global , Дата доступу до ресурсу: 20.11.2020

7. TrashOut. [Електронний ресурс] // Trash Out. – Режим доступу до ресурсу: https://trashout.ngo, Дата доступу до ресурсу: 20.11.2020

8. Spring Framework [Електронний ресурс] // Spring. – Режим доступу до ресурсу: https://spring.io/projects/spring-framework, Дата доступу до ресурсу: 20.11.2020

9. Docker [Електронний ресурс] // Docker. – Режим доступу до ресурсу: https://docker.com, Дата доступу до ресурсу: 20.11.2020

10. Flutter [Електронний ресурс] // Flutter. – Режим доступу до ресурсу: https://flutter.dev, Дата доступу до ресурсу: 20.11.2020

11. Geocoding API Overview [Електронний ресурс] // Google Cloud Platform. – Режим доступу до ресурсу: https://developers.google.com/maps/documentation/geocoding/overview, Дата доступу до ресурсу: 20.11.2020

12. Maps Static Overview [Електронний ресурс] // Google Cloud Platform. – Режим доступу до ресурсу: https://developers.google.com/maps/documentation/maps-static/overview, Дата доступу до ресурсу: 20.11.2020

13. Geocoding Api Key [Електронний ресурс] // Google Cloud Platform. – Режим доступу до ресурсу: https://developers.google.com/maps/documentation/geocoding/get-api-key, Дата доступу до ресурсу: 20.11.2020

14. Places API Overview [Електронний ресурс] // Google Cloud Platform. – Режим доступу до ресурсу: https://developers.google.com/places/web-service/overview, Дата доступу до ресурсу: 20.11.2020

15. Spring Data JPA Overview [Електронний ресурс] // Spring. – Режим доступу до ресурсу: https://spring.io/projects/spring-data-jpa, Дата доступу до ресурсу: 20.11.2020

16. PostgreSQL Docs [Електронний ресурс] // PostgreSQL. – Режим доступу до ресурсу: https://postgresql.org/docs, Дата доступу до ресурсу: 20.11.2020

17. The ultimate guide to mobile application architecture [Електронний ресурс] // AppVelocity. – Режим доступу до ресурсу: https://appvelocity.ca/guide-mobile-application-architecture, Дата доступу до ресурсу: 20.11.2020

18. JSON Overview [Електронний ресурс] // W3 Schools – Режим доступу: https:/w3.schools.com/js/js_json_intro.asp, Дата доступу до ресурсу: 20.11.2020

19. Tomcat [Електронний ресурс] // Tomcat – Режим доступу: http://tomcat.apache.org, Дата доступу до ресурсу: 20.11.2020

20. What is ORM and why should you use it? [Електронний ресурс] // ORM – Режим доступу: http://blog.bitsrc.io/what-is-an-orm-and-why-you-should-use-it-b2b6f75f5e2a, Дата доступу до ресурсу: 20.11.2020

21. Каленчук-Порханова Ж., Мовчан М., Поліщук В. Про актуальність моніторингу навколишнього середовища // Рідна природа.- 2002 – №2. – С. 12-14.

22. Качество и мониторинг окружающей природной среды // Безопасность жизнедеятельности: учебник / под ред. Э.А. Арустамова. – М.: Вагриус, 2003. – 533

23. Технології створення мобільних додатків [Електронний ресурс]. – Режим доступу: http://group-global.org/ru/publication/63343-tehnologii- sozdaniya-mobilnyh-prilozheniy, Дата доступу до ресурсу: 20.11.2020

24. AWS Amazon S3 [Електронний ресурс]. – Режим доступу: https://aws.amazon.com/aws/s3 , Дата доступу до ресурсу: 20.11.2020

25. Положення про Державну систему моніторингу довкілля. Постанова КМУ від 30.03.1998 р.-№391

26. Національна Доповідь про стан навколишнього середовища в Україні у 2000р. // Міністерствово екології та природних ресурсов України; Відп. за вип. О. Величко; Уклад. В. Романчук. – К., 2001. – 184 с.

27. Моделювання і прогнозування стану довкілля: підруч. / В. І. Лаврик, В. М. Боголюбов, Л. М. Полєтаєва та ін.; за ред. д.т.н. В. І. Лаврика. – К.: Академія, 2010.- 400 с.

28. Мытников А.Н., Мытникова Е.А., Кузнецова Л.Н., Солин С.Ю. Технологии разработки мобильных приложений // Теория и практика современной науки. – 2016. – № 4 (10). – С. 504-507.

29. Поліція планує запровадити в Україні мобільний додаток для миттєвого повідомлення про правопорушення. [Електронний ресурс] // NV.ua – Режим

доступу: https://nv.ua/ukraine/events/politsija-planiruet-vnedrit-v-ukraine-mobilnoe-prilozhenie-dlja-mgnovennogo-soobshchenija-o-pravonarushenijah-92606.html, Дата доступу до ресурсу: 20.11.2020

30. Скиба Ю. А. Моніторинг довкілля: навч. посібн. / Ю. А. Скиба, О.М. Лазебна; рец.: А. П. Галкін [та ін.]. – К. : Каравела, 2013. – 216с.

ДОДАТКИ

ДОДАТОК Д1

МОБІЛЬНИЙ ДОДАТОК ДЛЯ ЕКОМОНІТОРИНГУ ДОВКІЛЛЯ

Лістинг програми

Аркушів 15

Київ – 2020

MonitorApplication.java

```
package com.eco.monitor;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class MonitorApplication {

	public static void main(String[] args) {
		SpringApplication.run(MonitorApplication.class, args);
	}
}
```

PlaceResource.java

```
package com.eco.monitor.resource;

import com.eco.monitor.dto.IncidentDto;
import com.eco.monitor.dto.PlaceDto;
import com.eco.monitor.request.UpdatePlaceRequest;
import com.eco.monitor.service.IncidentService;
import com.eco.monitor.service.TokenService;
import com.eco.monitor.util.TokenUtil;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
```

```
import javax.servlet.http.HttpServletRequest;
import java.util.List;

@RestController
@RequestMapping(path = "/places")
public class PlaceResource {

    @Autowired
    public TokenService tokenService;

    @Autowired
    public IncidentService incidentService;

    @PostMapping
    public ResponseEntity createPlace(
            @RequestBody UpdatePlaceRequest request,
            HttpServletRequest httpServletRequest
    ) throws Exception {
        String token =
TokenUtil.resolveToken(httpServletRequest).orElseThrow(Exception::new);
        Integer userId = tokenService.decodeJWT(token);

        PlaceDto placeDto = new PlaceDto();
        placeDto.setAddress(request.getAddress());
        placeDto.setLongitude(Double.parseDouble(request.getLongitude()));
```

```
        placeDto.setLatitude(Double.parseDouble(request.getLatitude()));

        IncidentDto incidentDto = new IncidentDto();
        incidentDto.setTitle(request.getTitle());
        incidentDto.setCategory(request.getCategory());
        incidentDto.setDescription(request.getDescription());
        incidentDto.setPlace(placeDto);
        incidentDto.setImageUrl(request.getImageUrl());
        incidentDto.setUserId(userId);
        incidentService.createIncident(incidentDto);
        return ResponseEntity.ok().build();
    }

    @GetMapping
    public ResponseEntity<List<IncidentDto>> getIncidents() {
        return ResponseEntity.ok().body(incidentService.getIncidents());
    }

    @GetMapping(path = "/my")
    public ResponseEntity<List<IncidentDto>> getMyIncidents(@RequestParam
Integer userId) {
        return ResponseEntity.ok().body(incidentService.getMyIncidents(userId));
    }

    @GetMapping(path = "/{incidentId}")
```

```
    public ResponseEntity<IncidentDto> getIncident(@PathVariable Integer
incidentId) throws Exception {
        return ResponseEntity.ok().body(incidentService.getIncident(incidentId));
    }
}
```

UsersResource.java

```
package com.eco.monitor.resource;


import com.eco.monitor.dto.UserDto;
import com.eco.monitor.request.SignUpUserRequest;
import com.eco.monitor.request.SignUpUserResponse;
import com.eco.monitor.service.TokenService;
import com.eco.monitor.service.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;


@RestController
@RequestMapping(path = "/users")
public class UsersResource {


    @Autowired
    public UserService userService;
```

```
    @Autowired
    public TokenService tokenService;


    @PostMapping(path = "/sign-up")
    public ResponseEntity<SignUpUserResponse> signUpUser(@RequestBody
SignUpUserRequest request) throws Exception {
        UserDto user = userService.addUser(request.getEmail(),
request.getPassword());


        String token = tokenService.createJWT(user.getId().toString());
        System.out.println("NEW USER WITH ID = " + user.getId() + " WAS
REGISTERED");
        return ResponseEntity.ok(new SignUpUserResponse(token, user.getId()));
    }


    @PostMapping(path = "/sign-in")
    public ResponseEntity<SignUpUserResponse> signInUser(@RequestBody
SignUpUserRequest request) throws Exception {
        UserDto user = userService.getUser(request.getEmail(),
request.getPassword());


        String token = tokenService.createJWT(user.getId().toString());
        System.out.println("NEW USER WITH ID = " + user.getId() + " WAS
LOGGED IN");
        return ResponseEntity.ok(new SignUpUserResponse(token, user.getId()));
    }
```

}

IncidentService.java

```
package com.eco.monitor.service;

import com.eco.monitor.dto.IncidentDto;
import com.eco.monitor.entity.Incident;
import com.eco.monitor.entity.Place;
import com.eco.monitor.entity.User;
import com.eco.monitor.repository.IncidentRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

import static com.eco.monitor.converter.IncidentConverter.toIncidentDto;
import static com.eco.monitor.converter.IncidentConverter.toIncidentDtoList;

@Service
public class IncidentService {

    @Autowired
    private IncidentRepository incidentRepository;

    public void createIncident(IncidentDto incidentDto) {
        Place place = new Place();
        place.setAddress(incidentDto.getPlace().getAddress());
```

```
        place.setLatitude(incidentDto.getPlace().getLatitude());
        place.setLongitude(incidentDto.getPlace().getLongitude());

        Incident incident = new Incident();
        incident.setPlace(place);
        incident.setCategory(incidentDto.getCategory());
        incident.setDescription(incidentDto.getDescription());
        incident.setTitle(incidentDto.getTitle());
        incident.setImage(incidentDto.getImageUrl());
        incident.setUser(new User(incidentDto.getUserId()));

        incidentRepository.save(incident);
    }

    public IncidentDto getIncident(Integer incidentId) throws Exception {
        return
toIncidentDto(incidentRepository.findById(incidentId).orElseThrow(Exception::n
ew));
    }

    public List<IncidentDto> getIncidents() {
        return toIncidentDtoList((List<Incident>) incidentRepository.findAll());
    }

    public List<IncidentDto> getMyIncidents(Integer userId) {
        return toIncidentDtoList(incidentRepository.findByUserId(userId));
```

```
    }
}
```

TokenService.java

```
package com.eco.monitor.service;

import com.eco.monitor.config.ConfigurationManager;
import io.jsonwebtoken.Claims;
import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.SignatureAlgorithm;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import javax.crypto.spec.SecretKeySpec;
import javax.xml.bind.DatatypeConverter;
import java.security.Key;
import java.util.Date;

@Service
public class TokenService {

    @Autowired
    private ConfigurationManager configurationManager;

    public String createJWT(String userId) {

        //The JWT signature algorithm we will be using to sign the token
```

```
        SignatureAlgorithm signatureAlgorithm = SignatureAlgorithm.HS256;


        long nowMillis = System.currentTimeMillis();
        Date now = new Date(nowMillis);


        byte[] apiKeySecretBytes =
DatatypeConverter.parseBase64Binary(configurationManager.getJwtSecret());
        Key signingKey = new SecretKeySpec(apiKeySecretBytes,
signatureAlgorithm.getJcaName());


        return Jwts.builder()
                .claim("scope", "user")
                .setIssuedAt(now)
                .setIssuer(userId)
                .setExpiration(new Date(nowMillis + 7200000))
                .signWith(signatureAlgorithm, signingKey)
                .compact();
    }


    public Integer decodeJWT(String jwt) {
        Claims claims = Jwts.parser()

.setSigningKey(DatatypeConverter.parseBase64Binary(configurationManager.getJ
wtSecret()))
                .parseClaimsJws(jwt).getBody();
        return Integer.valueOf(claims.getIssuer());
    }
```

}

Incident.java

```
package com.eco.monitor.entity;


import javax.persistence.*;


@Entity
@Table(name = "incidents")
public class Incident {


    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;
    private String title;
    private String description;
    private String category;
    private String image;


    @OneToOne(cascade = {CascadeType.PERSIST, CascadeType.MERGE})
    private Place place;


    @ManyToOne(fetch = FetchType.EAGER)
    @JoinColumn(name = "userId", referencedColumnName = "id", updatable =
false, nullable = false)
    private User user;
```

```
public Integer getId() {
    return id;
}

public void setId(Integer id) {
    this.id = id;
}

public String getTitle() {
    return title;
}

public void setTitle(String title) {
    this.title = title;
}

public String getDescription() {
    return description;
}

public void setDescription(String description) {
    this.description = description;
}

public String getCategory() {
```

```
    return category;
}

public void setCategory(String category) {
    this.category = category;
}

public Place getPlace() {
    return place;
}

public void setPlace(Place place) {
    this.place = place;
}

public String getImage() {
    return image;
}

public void setImage(String image) {
    this.image = image;
}

public User getUser() {
    return user;
```

```
    }

    public void setUser(User user) {
        this.user = user;
    }
}
```

User.java

```
package com.eco.monitor.entity;

import javax.persistence.*;

@Entity
@Table(name = "users")
public class User {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;
    private String email;
    private String password;

    public User() {
    }

    public User(Integer id) {
        this.id = id;
```

```
}

public Integer getId() {
    return id;
}

public void setId(Integer id) {
    this.id = id;
}

public String getEmail() {
    return email;
}

public void setEmail(String email) {
    this.email = email;
}

public String getPassword() {
    return password;
}

public void setPassword(String password) {
    this.password = password;
}
```

}